TOPICS ON THE LONGEST COMMON SUBSEQUENCES: SIMULATIONS, COMPUTATIONS, AND VARIANCE

A Dissertation
Presented to
The Academic Faculty

By

Qingqing Liu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mathematics

Georgia Institute of Technology

December 2018

TOPICS ON THE LONGEST COMMON SUBSEQUENCES: SIMULATIONS, COMPUTATIONS, AND VARIANCE

Approved by:

Dr. Christian Houdré, Advisor
School of Mathematics
Georgia Institute of Technology

Dr. Michael Damron
School of Mathematics
Georgia Institute of Technology

Dr. Vladimir Koltchinskii
School of Mathematics
Georgia Institute of Technology

Dr. Yajun Mei
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Mayya Zhilova
School of Mathematics
Georgia Institute of Technology

Date Approved: Nov 1, 2018

To Rundong and Anya.

ACKNOWLEDGEMENTS

First or all, I would like to thank my advisor Prof. Christian Houdré. He taught me how to think independently and conduct meticulous research. He provided me inspiring insights when I met difficulties in research. He also provided valuable help in my academic writing.

I would like to thank Prof. Michael Damron, Prof. Vladimir Koltchinskii, Prof. Yajun Mei, Prof. Mayya Zhilova for serving in my thesis committee and the insights they provided.

I would like to thank Prof. Karim Lounici. He, Prof. Houdré and I had several enjoyable discussions together, where he suggested some interesting applications of my study in simulations, computations, and statistics for longest common subsequences, which later developed into Section 3.2 of this dissertation. I would like to thank Klara Grodzinsky for her supportive teaching arrangement.

I would like to thank the people who maintained the PACE (Partnershipfor an Advanced Computing Environment) clusters, where most experiments in this dissertation were performed.

I would like to thank Jing Hu, Tongzhou Chen, Longmei Shu, Gagik Amirkhanyan, Juntao Duan, Xiaolin Wang, Ruidong Wang, Yunlong He, Yuze Zhang, Fan Zhou, Yueqin Zhong, Yu Zhuo and Shanshan Ma for the helpful discussions and exciting moments.

Finally, I would like to thank my family for their love and support. I own innumerable thanks to my parents for their encouragement and support, to my husband for his tremendous love and selfless help, and to my dearest daughter, for bringing so much happiness and special moments into my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The study of the longest common subsequences (LCSs) of two random words/strings is classical in computer science and bioinformatics. A problem of particular probabilistic interest is to determine the limiting behavior of the expectation and variance of the length of the LCSs, as the length of the random words grows without bound. This dissertation studies this problem using both Monte-Carlo simulation and theoretical analysis. The specific questions studied here include estimating the growth order of the variance, LCSs based hypothesis testing methods for sequences similarity, theoretical upper bounds on the Chvátal-Sankoff constant of multiple sequences, and theoretical growth order of the variance when the two random words have asymmetrical distributions.

CHAPTER 1

INTRODUCTION AND BACKGROUND

The study of the longest common subsequences (LCSs) of two random words/strings is a classical topic in computer science and bioinformatics, since this length is often used to measure the similarity/dissimilarity of the two words/strings. Let $\boldsymbol{X} = (X_i)_{1\le i\le n_1}$ and $\boldsymbol{Y} = (Y_j)_{1\le j\le n_2}$ be two independent finite sequences of i.i.d. random variables taking values in the alphabet $\mathcal{A} = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$. Then, a common subsequence $\boldsymbol{Z}$ of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is a sequence such that

$$Z_1 = X_{i_1} = Y_{j_1}, \cdots, Z_\ell = X_{i_\ell} = Y_{j_\ell},$$

for some $1 \le i_1 < \cdots < i_\ell \le n_1$, $1 \le j_1 < \cdots < j_\ell \le n_2$.

A longest common subsequence is a common subsequence of maximum length. We denote the length of a longest common subsequence by $LC(\boldsymbol{X}, \boldsymbol{Y})$, and by LC_n if the two sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ are of the same length n.

For finite sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$, finding the length of the longest common subsequences can be done, via dynamic programming, in polynomial time. A question of particular probabilistic interest is then to determine the limiting behavior of the expected length of the LCS as n grows without bound.

Chvátal and Sankoff [5] were the first to study this problem, and proved that the limit

$$\gamma_m^* = \lim_{n\to\infty} \frac{\mathbb{E}LC_n}{n},$$

exists, where m is the size of the alphabet, and the expectation is taken assuming the sequences are i.i.d. generated and are independent of each other. Their proof was based on Fekete's lemma [10] and the fact that $\mathbb{E}\,LC_n$ is superadditive, i.e., that $\mathbb{E}\,LC_{k+\ell} \ge \mathbb{E}\,LC_k + \mathbb{E}\,LC_\ell$, $k, \ell \in \mathbb{N}$.

Since it has been shown to exist, much effort has been put into finding the exact values of γ_m^*, but this remains unknown up to this point, and numerous methods giving bounds for γ_m^* have been developed.

Starting with Chvátal and Sankoff [5], theoretical bounds are given for the binary uniform case. These are summarized in Table 1.1.

Table 1.1: Theoretical Bounds for γ_2^*

	lower bound	upper bound
Chvátal and Sankoff [5, 6]	0.727273	0.86660
Deken [8, 9]	0.7615	0.8575
Dančík [7]	0.773911	0.837623
Lueker [23]	0.788071	0.826280
Lueker (our run)*	0.789883	0.825101

* We run Lueker's program using better parameters with better computing resources.

Writing $\mathbb{E}\, LC_n$ as

$$\mathbb{E}\, LC_n = \frac{1}{m^{2n}} \sum_{\boldsymbol{X},\boldsymbol{Y} \in \mathcal{A}^n} LC(\boldsymbol{X}, \boldsymbol{Y}), \tag{1.1}$$

Chvátal and Sankoff [5] computed its explicit form for $1 \le n \le 5$. However, as n grows, the number of terms in (1.1) grows exponentially and quickly becomes hard to estimate, even for $m = 2$. To obtain an upper bound on γ_2^*, all the methods in Table 1.1 are built on the following basis. Let $G(n, \ell)$ be the number of string pairs that have an LCS of length at least ℓ, then one can also write $\mathbb{E}\, LC_n$ as

$$\mathbb{E}\, LC_n = \frac{1}{m^{2n}} \sum_{\ell=1}^{n} G(n, \ell). \tag{1.2}$$

Next, a strategy to find an upper bound on γ_m^* is to allow for some overcounting in $G(n, \ell)$ in return for more tractable computation. The improvements of the upper bound in Table 1.1 are all about reducing the overcounting. A key theorem that directly connects estimation of $G(n, \ell)$ with upper bound of γ_m^* is given by

Theorem 1.1 (Chvátal and Sankoff [5]) *If for some alphabet of size m, and for some*

$y \in (0, 1)$,

$$G(n, yn) = o(m^{2n}),$$

as $n \to \infty$, *then* $\gamma_m^* \leq y$.

Various combinatorial techniques have been used to estimate $G(n, \ell)$, from estimates on binomial coefficients to the use of generating functions. Some methods [7, 23] even incorporate finite state machines. Unlike upper bounds, lower bound estimates do not have a common basis and the current state-of-the-art method [23] uses a dynamic programming approach.

After the expectation, the study of the variance of LC_n naturally comes next. The study of this aspect, however, is less complete. Steele [26] showed the generic upper bound:

$$\mathrm{Var}(LC_n) \leq n \left(1 - \sum_{i=1}^{m} p_i^2\right), \tag{1.3}$$

where $p_i = \mathbb{P}(X_1 = \alpha_i), \alpha_i \in \mathcal{A} = \{\alpha_1, \cdots, \alpha_m\}$.

For lower bounds, the linear order results are only proved in various biased instances ([20], [15], [16], [21], [11], [1] $\cdots$). For example, [15] assumes that one of the letters has a significantly higher probability of appearing than any of the other letters in the alphabet, while [3] assumes that one of the two sequences has a binary alphabet while the other has a trinary one. This dissertation (Chapter 4) extends the result of [3] by removing the binary/trinary assumption and provides precise estimates allowing us to go beyond the uniform case and to also deal with central moments.

The rest of the thesis is arranged as follows. In Chapter 2 we discuss current state-of-the-art algorithms for computing the length of the longest common subsequence. This is important, in our following chapters, in order to simulate sequences. Chapter 3, the content of which is published in [22], is focused on the case where all the sequences are independent, with values taken i.i.d. from the alphabet $\mathcal{A} = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$. We study the variance of the length of the longest common subsequence, via extensive simulations, and explore the

use of hypothesis testing for the similarity of two sequences. We also obtain upper bounds when more than two sequences are involved. In Chapter 4, the content of which will be published as [13], we look at the asymmetric case, with each value of one sequence taken, with positive probability, from $\{\alpha_1, \alpha_2, \cdots \alpha_m, \alpha_{m+1}\}$, while for the other sequence α_{m+1} is taken with probability zero. In this case, we prove that the order of the variance is linear. Finally, we conclude this thesis in Chapter 5 by also briefly presenting some open problems for potential future work.

CHAPTER 2

ALGORITHMS FOR LONGEST COMMON SUBSEQUENCES

Let $\boldsymbol{X} = (X_i)_{1\le i\le n_1}$ and $\boldsymbol{Y} = (Y_i)_{1\le j\le n_2}$ be two finite sequences, with letters from $\mathcal{A} = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$. Much effort has been put into finding the length of the longest common subsequence of $\boldsymbol{X}$ and $\boldsymbol{Y}$. Let us summarize some of the results.

2.1 Dynamic Programming

The first approach using dynamic programming was proposed by Wagner and Fisher [27] in 1974. In the case of the longest common subsequence, we have

$$
\begin{aligned}
&LC(\boldsymbol{X}[1:i], \boldsymbol{Y}[1:j]) \\
&= \begin{cases} 0 & \text{if } i = 0, \text{ or } j = 0 \\ LC(\boldsymbol{X}[1:i-1], \boldsymbol{Y}[1:j-1]) + 1 & \text{if } X_i = Y_j \\ \max\{LC(\boldsymbol{X}[1:i-1], \boldsymbol{Y}[1:j]), LC(\boldsymbol{X}[1:i], \boldsymbol{Y}[1:j-1])\} & \text{if } X_i \neq Y_j. \end{cases}
\end{aligned} \tag{2.1}
$$

Here, $\boldsymbol{X}[1:i]$ denotes the subsequence $X_1X_2\cdots X_i$, and similarly $\boldsymbol{Y}[1:j]$ denotes the subsequence $Y_1Y_2\cdots Y_j$.

Using the recursive relation in (2.1), we can compute the the length of the longest common subsequence using dynamic programming as described in Algorithm 1. If one is interested in finding the longest common subsequence, we can backtrack the dynamic programming table (the D matrix generated in Algorithm 1). The main idea is that if $X_i = Y_j$, then (X_i, Y_j) must be a matching pair in the LCS. If $X_i \neq Y_j$, one can check whether $D(i-1, j)$ or $D(i, j-1)$ gives the larger value, and then move forward to the larger position.

The time complexity and space complexity of Algorithm 1 are both $O(n_1n_2)$, which is inefficient when the two sequences are very long, and when Algorithm 1 needs to be

Algorithm 1 Wagner and Fischer [27]

```
1:  procedure LC(X, Y)
2:      n1 = |X|, n2 = |Y|
3:      D(0, 0) = 0
4:      for i = 1 to n1 do
5:          D(i, 0) = 0
6:      end for
7:      for j = 1 to n2 do
8:          D(0, j) = 0
9:      end for
10:     for i = 1 to n1 do
11:         for j = 1 to n2 do
12:             if X_i = Y_j then
13:                 D(i, j) = D(i − 1, j − 1) + 1
14:             else
15:                 D(i, j) = max{D(i − 1, j), D(i, j − 1)}
16:             end if
17:         end for
18:     end for
19:     return D(n1, n2)
20: end procedure
```

repeated multiple times.

Plenty of effort has been made to reduce the time and space complexity of Wagner and Fischer's algorithm. Hirschberg[12] improved the space complexity to be linear in 1975, and Hunt and Szymanski[17] improved the time complexity to $O((r + n) \log n)$ in 1977. Here, r is the number of match pairs, i.e., the number of pairs in $\{(i, j) : X[i] = Y[j]\}$, and it is assumed $n = n_1 = n_2$. Barš studied many LCS algorithms and their time and space complexity in his thesis [2]. From his experimental results, we find that the Kuo-Cross algorithm [19] and WMMM [28] both use memory efficiently, and in particular the Kuo-Cross algorithm is fast for large alphabet sizes and WMMM is fast for very small (less than four) alphabet sizes.

Most of the improved algorithms were still based on the dynamic programming approach. Some popular acceleration strategies were (1) to replace $LC(\boldsymbol{X}, \boldsymbol{Y})$ with an equivalent variable that is relatively more efficient to compute and (2) to avoid unnecessary computations

when filling the dynamic programming table. Both Kuo-Cross and WMMM algorithms described in the following sections used these two strategies.

2.2 Kuo-Cross Algorithm

2.2.1 The original Kuo-Cross algorithm

The Kuo-Cross algorithm [19] is based on an improvement of the Hunt-Szymanski algorithm [17], where $n = n_1 = n_2$ is assumed. Hunt and Szymanski defined a $THRESHOLD$ array to be

$$T_{i,k} = \min\{j : X[1:i] \text{ and } Y[1:j] \text{ contain a common subsequence of length } k.\}$$

They also proved the recursive formula for $T_{i,k}$:

$$T_{i,k} = \begin{cases} \min\{j : X[i] = Y[j] \text{ and } T_{i-1,k-1} < j \leq T_{i-1,k}\} \\ T_{i-1,k} \text{ if no such } j \text{ exists.} \end{cases}$$

From the definition of $T_{i,k}$, the largest k such that $T_{n,k}$ is defined is the length of the LCS.

A naive implementation of the above dynamic programming formula would take $O(n^2)$ time. By using a list array called *MATCHLIST* which recorded lists of positions of matched letters of $X[i]$ in $\boldsymbol{Y}$, Hunt and Szymanski accelerated the algorithm by only updating $T_{i,k}$ that were different from $T_{i-1,k}$, with a resulting time complexity $O((r+n)\log n)$, where r is the number of match pairs, and n is the length of the sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$. Kuo and Cross improved their algorithm by putting *MATCHLIST* in increasing order, and further reducing unnecessary updates of the *THRESHOLD* array, with a resulting time complexity $O(r + n(LCS(\boldsymbol{X}, \boldsymbol{Y}) + \log n))$ and space complexity $O(n + r)$.

Kuo-Cross algorithm is described as in Algorithm 2. It is efficient when the alphabet size is large.

Algorithm 2 Kuo-Cross algorithm [19]

procedure LC($\boldsymbol{X}, \boldsymbol{Y}$)
    # Step 1: Build Linked Lists
    for $i = 1$ to n **do**
        set $MATCHLIST[i] =< j_1, j_2, \cdots, j_p >$ such that
        $j_1 < j_2 < \cdots < j_p$ and $X[i] = Y[j_q]$ for $1 \leq q \leq p$
    end for
    # Step 2: Initialize *THRESH* Array
    $THRESH[0] = 0$
    for $i = 1$ to n **do**
        $THRESH[i] = n + 1$
    end for
    # Step 3: Compute Successive *THRESH* Values
    for $i = 1$ to n **do**
        temp = 0, k = 0
        for j in $MATCHLIST[i]$ **do**
            if $j > temp$ **then**
                while $j > THRESH[k]$ **do**
                    $k = k + 1$
                end while
                $temp = THRESH[k]$
                $THRESH[k] = j$
            end if
        end for
    end for
    return the largest k such that $THRESH[k] \neq n + 1$
end procedure

2.2.2 The improved Kuo-Cross algorithm

In Step 1 of Algorithm 2, Kuo and Cross sorted the sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ to obtain the *MATCHLIST*, which took $O(n \log n)$ time. However, we can improve this step to $O(n)$ using Algorithm 3. Without loss of generality, we can assume the alphabet consists of $\mathcal{A} = \{1, 2, \cdots, m\}$, otherwise we can use a hash table to map each letter to an integer index in $O(1)$ time. In practice, all the linked lists in Algorithm 2 can be compactly encoded in an array of length n.

Algorithm 3 Improved Kuo-Cross algorithm

```
1:  procedure (# Step 1′)
2:      # Step 1′ : Build Linked Lists
3:      Initialize ALIST to be an array of k empty lists.
4:      for j = 1 to n do
5:          Append j to the list stored at ALIST[Y[j]]
6:      end for
7:      for i = 1 to n do
8:          MATCHLIST[i] = ALIST[X[i]]
9:      end for
10: end procedure
```

With our improvement, the new algorithm will have time complexity $O(r+nLCS(\boldsymbol{X}, \boldsymbol{Y}))$, where r denotes the number of match pairs.

2.3 WMMM Algorithm

Another algorithm we used in the simulations in later sections is the WMMM algorithm by Wu, Manber, Myers, and Miller[28]. In their algorithm, they employed the relations between the shortest edit script and the longest common subsequence. In this section, we assume $n_1 = |\boldsymbol{X}| \leq n_2 = |\boldsymbol{Y}|$ without loss of generality, and let $\Delta = n_2 - n_1$.

For two sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$, a list of insert or delete instructions needed to edit $\boldsymbol{X}$ to $\boldsymbol{Y}$ is an edit script, and the shortest edit script is an edit script of minimum length. Let P be the number of delete instructions in the shortest edit script. Clearly, the length of LCS is $n_1 - P$.

Let $V(i, j)$ denote the number of deletions in the shortest edit script for $\boldsymbol{X}[1 : i]$ and $\boldsymbol{Y}[1 : j]$, and we can obtain a lower bound on P assuming the final edit script will first edit $\boldsymbol{X}[1 : i]$ into $\boldsymbol{Y}[1 : j]$, as following:

$$P(i, j) = \begin{cases} V(i, j) & \text{if } j - i \leq \Delta \\ V(i, j) + (j - i) - \Delta & \text{if } j - i > \Delta. \end{cases}$$

Next they defined

$$fp(k, p) = \max\{j : P(j - k, j) = p\},$$

and

$$FP(p) = \{(j - k, j) : j = fp(k, p) \text{ and } -p \leq k \leq p + \Delta\}.$$

In their algorithm, they iteratively computed the set $FP(p)$ from the set $FP(p - 1)$ until (n_1, n_2) was in $FP(p)$ where both P and the length of LCS were known. The authors showed that using such constructions, the algorithm only needed to examine the (i, j) pairs that lie in the band $\{(i, j) : -P \leq j - i \leq \Delta + P\}$, instead of the whole dynamic programming table.

The worst-case running time for the WMMM algorithm is $O(n_2 P)$, and the expected running time is $O(n_2 + P(\Delta + 2P))$. The space complexity is $O(n_1 + n_2)$.

The WMMM algorithm is described in Algorithm 4. It is fast for small (less than four) alphabet sizes.

Algorithm 4 WMMM algorithm

```
1: procedure LC(X, Y)
2:     n_1 = |X|, n_2 = |Y|
3:     f_p[−n_1, ··· , n_2] = −1
4:     Δ = n_2 − n_1
5:     p = −1
6:     while f_p(Δ) ≠ N do
7:         p = p + 1
8:         for k = −p to Δ − 1 do
9:             f_p[k] = SNAKE(k, max(f_p[k − 1] + 1, f_p[k + 1]))
10:        end for
11:        for k = Δ + p to Δ + 1 do
12:            f_p[k] = SNAKE(k, max(f_p[k − 1] + 1, f_p[k + 1]))
13:        end for
14:        f_p[Δ] = SNAKE(k, max(f_p[k − 1] + 1, f_p[k + 1]))
15:    end while
16:    return n_1 − p
17: end procedure
18:
19: procedure SNAKE(k, y)
20:    n_1 = |X|, n_2 = |Y|
21:    x = y − k
22:    while x < n_1, y < n_2, and X[x + 1] = Y[y + 1] do
23:        x = x + 1
24:        y = y + 1
25:    end while
26:    return y
27: end procedure
```

CHAPTER 3

SIMULATIONS RESULTS AND THEORETICAL BOUNDS

In this chapter, we first study the statistical behavior in mean and variance of the length of the longest common subsequences using a Monte Carlo approach from which we then develop a hypothesis testing method for sequences similarity. Later, theoretical upper bounds are obtained for the Chvátal-Sankoff constant of multiple sequences.

3.1 Monte Carlo Simulation for the Variance

As previously mentioned, the theoretical study of the variance of the length of LCSs is less complete. A general linear upper bound has been obtained in [26]. Lower bounds, also of linear order, have been proven in various biased instances ([20], [15], [16], [21], [11], [1] $\cdots$). But the uniform i.i.d. case is still unknown. In [4], it is observed through Monte Carlo simulation, with n up to $20,000$, that the order of the variance of the length of the LCSs of binary random words is at least of order $n^{2\omega'}$, where $\omega' = 0.418 \pm 0.005$. Our simulation shows that when n becomes larger, such deviation also becomes larger and the variance tends to have order n.

3.1.1 Problem Description

Given two sequences $\boldsymbol{X} = X_1 \cdots X_n$ and $\boldsymbol{Y} = Y_1 \cdots Y_n$ having the same length, where $X_i, Y_i \in \mathcal{A}$ and where again $\mathcal{A}$ is the alphabet, we explore, by Monte Carlo method, the asymptotic behavior of Var LC_n when n grows large.

To perform Monte Carlo simulations, we need to select an algorithm to compute the length of the LCSs. The dynamic programming algorithm is classical but not efficient enough, as discussed in Section 2.1. Since our experiments are only for $|\mathcal{A}| = 2$ or $|\mathcal{A}| = 4$, we choose to use the WMMM algorithm [28], which according to [2] is very efficient in

time and memory when $|\mathcal{A}|$ is small.

3.1.2 Experiment Setting

- The alphabet size is 2 ($|\mathcal{A}| = 2$);
- For each n we draw 10,000 random sample for Monte Carlo simulation.

3.1.3 Experiment Results

$\mathbb{P}(X_1 = 0) = 0.5,\ \mathbb{P}(X_1 = 1) = 0.5$

In this experiment, n ranges from 50,000:50,000:1,000,000. We plot Var LC_n against n under a log-log scale in Figure 3.1.

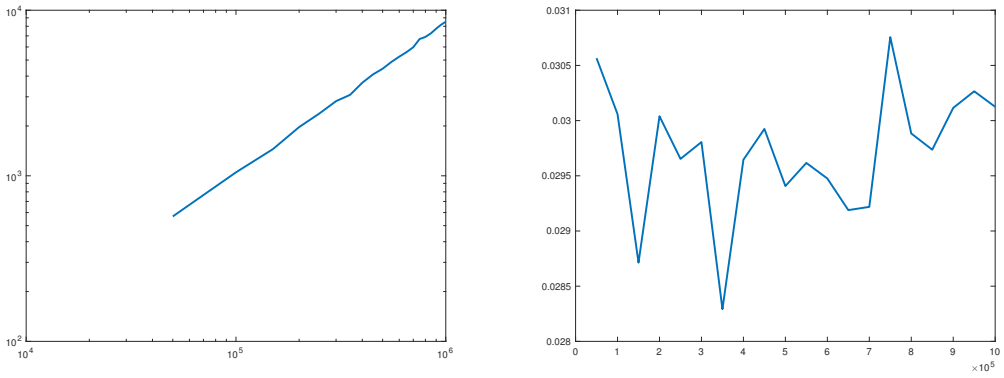

Figure 3.1: **Left**: log-log plot of Var LC_n versus n, **Right**: plot of Var $LC_n/n^{0.9086}$ versus n

We found the following relation between Var LC_n and n using linear regression

$$\text{Var } LC_n \approx 0.0297 n^{0.9086}.$$

$\mathbb{P}(X_1 = 0) = 0.1,\ \mathbb{P}(X_1 = 1) = 0.9$

In this experiment, n ranges from 50,000:50,000:1,000,000. We plot Var LC_n against n under a log-log scale in Figure 3.2.

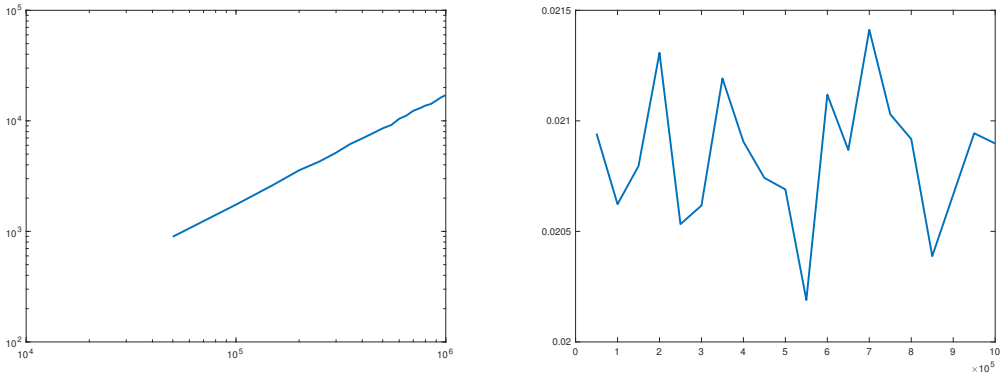

Figure 3.2: **Left**: log-log plot of Var LC_n versus n, **Right**: plot of Var $LC_n/n^{0.9855}$ versus n

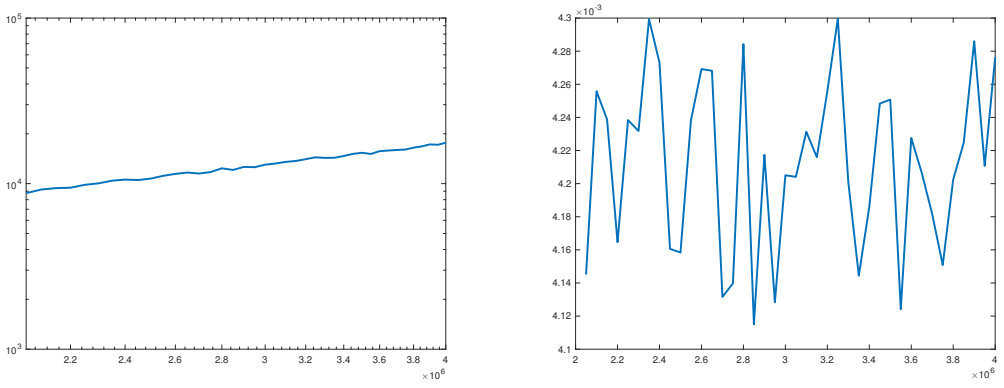

Figure 3.3: **Left**: log-log plot of Var LC_n versus n, **Right**: plot of Var $LC_n/n^{1.0021}$ versus n

We found the following relation between Var LC_n and n using linear regression

$$\text{Var } LC_n \approx 0.0208 n^{0.9855}.$$

$\mathbb{P}(X_1 = 0) = 0.01,\ \mathbb{P}(X_1 = 1) = 0.99$

In this experiment, n ranges from 2,050,000:50,000:4,000,000. We plot Var LC_n against n under a log-log scale in Figure 3.3.

We found the following relation between Var LC_n and n using linear regression

$$\text{Var } LC_n \approx 0.0042 n^{1.0021}.$$

In all cases, we conjecture that the order of variance of LC_n is:

$$VarLC_n \overset{asym}{\sim} cn,$$

where c is a small constant.

3.2 Hypothesis Testing for the Similarity of two Sequences

3.2.1 Testing Procedure

To test the similarity of two sequences, we propose the following hypothesis testing procedure. Assume we have two sequences $\boldsymbol{X} = X_1 \cdots X_n$ and $\boldsymbol{Y} = Y_1 \cdots Y_n$, both of length n, and then define the null and alternative hypothesis as

$$H_0 : \boldsymbol{X} \text{ and } \boldsymbol{Y} \text{ are i.i.d. uniformly generated}$$

$$H_a : \boldsymbol{X} \text{ and } \boldsymbol{Y} \text{ have high similarity.}$$

Based on the results of [14], we use the Z-test and the test statistic is

$$S = \frac{(LC_n)_{obs} - \mathbb{E}LC_n}{\sqrt{\text{Var}\, LC_n}}, \tag{3.1}$$

where $(LC_n)_{obs}$ is the observed length of the LCS of the two sequences being tested, while $\mathbb{E}LC_n$ and $\text{Var}\, LC_n$ are the expectation and variance of the length of the LCSs of two sequences, their values estimated by Monte Carlo simulation.

The paper [25] proposed a similarity score based on LCS for comparing two sequences without providing a hypothesis testing procedure, where the estimated LCS statistics were computed for n up to 1000. Below, we develop a hypothesis testing approach and conduct simulations for $n = 10,000$ and extensively verified the effectiveness of the testing method on synthetic sequences.

3.2.2 Experimental Verification

We conducted several experiments to verify the effectiveness of our testing procedure still using the WMMM algorithm. These experiments shares the following assumptions/parameters:

- The alphabet size is 4 ($|\mathcal{A}| = 4$);
- The two sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ have the same length ($|\boldsymbol{X}| = |\boldsymbol{Y}| = n$);
- The action of inserting a sequence $\boldsymbol{Z}$ into another sequence $\boldsymbol{X}$ is controlled by a parameter s. We divide $\boldsymbol{Z}$ into s equally long contiguous segments and $\boldsymbol{X}$ into $s + 1$ equally long contiguous segments, and then insert the s segments from $\boldsymbol{Z}$ into corresponding positions in the s gaps of $\boldsymbol{X}$, as illustrated in Figure 3.4. We denote this action as $\textsc{insert}(\boldsymbol{Z}, \boldsymbol{X}, s)$.

With $n = 1,000,000$, we randomly generated 529 pairs of $\boldsymbol{X}$ and $\boldsymbol{Y}$, and compute $\gamma_4^* \approx \overline{\text{LCS}(X,Y)}/n \approx 0.654$, $c \approx s^2(\text{LCS}(X,Y))/n \approx 0.0075$.

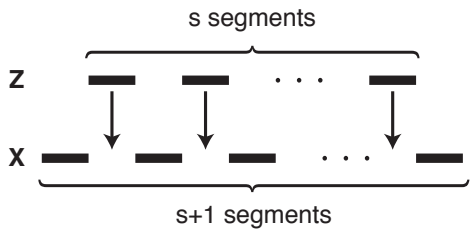

Figure 3.4: Inserting $\boldsymbol{Z}$ into $\boldsymbol{X}$.

We use $\alpha = 0.05$, $n = 10,000$ in our experiments. For each Monte Carlo simulation, we draw $10,000$ random samples.

Below are the experiment results.

Null Hypothesis

Here $\mathbb{P}(S \leq Z_\alpha) = 0.9893$, and the histogram of $((LC_n)_{obs} - \gamma_4^* n)/\sqrt{cn}$ is in Figure 3.5.

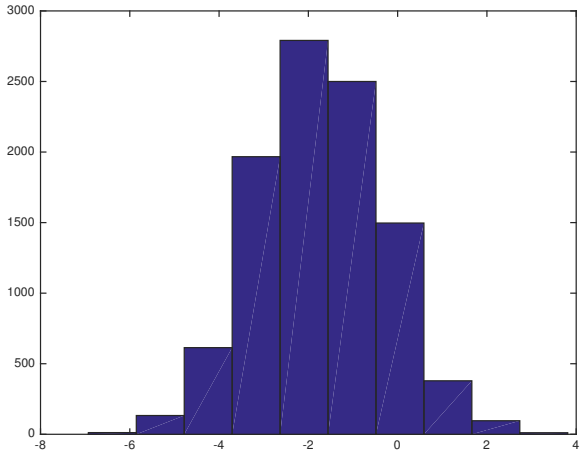

Figure 3.5: Histogram of $\frac{(LC_n)_{obs} - \gamma_4^* n}{\sqrt{cn}}$

Alternative Hypothesis (1)

H_a: We randomly generated two uniform i.i.d. sequences $\boldsymbol{X}'$, $\boldsymbol{Y}'$ of length n', and insert a sequence $\boldsymbol{Z}$ of length $n - n'$ into $\boldsymbol{X}'$ and $\boldsymbol{Y}'$, obtaining $\boldsymbol{X}$ and $\boldsymbol{Y}$. The results for $p = \mathbb{P}(S \leq Z_\alpha)$ are in Table 3.1.

Table 3.1: Results for $p = \mathbb{P}(S \leq Z_\alpha)$

n'	$n - n'$	p
9,000	1,000	0
9,300	700	0.2284
9,350	650	0.4286
9,400	600	0.6119
9,500	500	0.8541
9,900	100	0.9884

Alternative Hypothesis (2)

H_a: We randomly generated two uniform i.i.d. sequences $\boldsymbol{X}'$, $\boldsymbol{Y}'$ of length $n' = 5,000$, and inserted a sequence $\boldsymbol{Z}$ of length $n - n' = 5,000$ into $\boldsymbol{X}'$ and $\boldsymbol{Y}'$ obtaining $\boldsymbol{X}$ and $\boldsymbol{Y}$. The difference is now that each piece of the sequence $\boldsymbol{Z}$ has been inserted, with probability 0.8 into both $\boldsymbol{X}'$ and $\boldsymbol{Y}'$, with probability 0.1 into $\boldsymbol{X}'$ alone, and with probability 0.1 into $\boldsymbol{Y}'$ alone.

In this case, $\mathbb{P}(S \leq Z_\alpha) = 0$, and the histogram of $((LC_n)_{obs} - \gamma_4^* n)/\sqrt{cn}$ is in Figure 3.6.

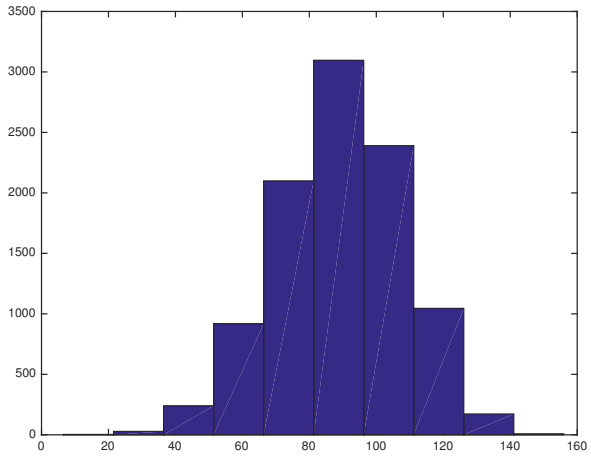

Figure 3.6: Histogram of $\frac{(LC_n)_{obs} - \gamma_4^* n}{\sqrt{cn}}$

Alternative Hypothesis (3)

H_a: We randomly generated two uniform i.i.d. sequences $\boldsymbol{X}'$, $\boldsymbol{Y}'$ of length $n' = 5,000$, and insert a sequence $\boldsymbol{Z}$ of length $n - n' = 5,000$ into $\boldsymbol{X}'$ and $\boldsymbol{Y}'$ obtaining $\boldsymbol{X}$ and $\boldsymbol{Y}$. This time, each piece of the sequence $\boldsymbol{Z}$ was inserted with probability 0.15 into both $\boldsymbol{X}'$ and $\boldsymbol{Y}'$, with probability 0.4 into $\boldsymbol{X}'$ alone, with probability 0.4 into $\boldsymbol{Y}'$ alone, and with probability 0.05 into neither $\boldsymbol{X}'$ nor $\boldsymbol{Y}'$.

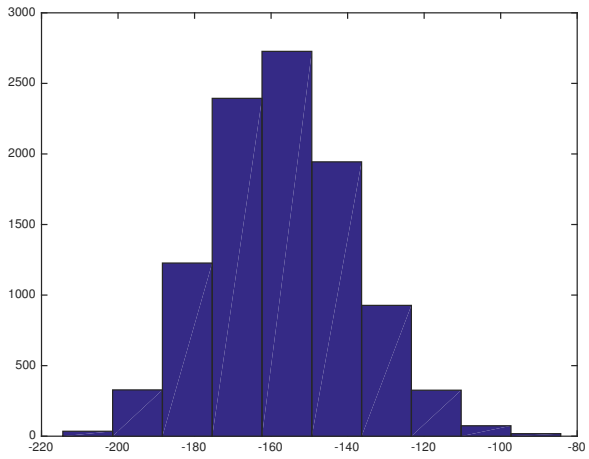

Figure 3.7: Histogram of $\frac{(LC_n)_{obs} - \gamma_4^* n}{\sqrt{cn}}$

In this case, $\mathbb{P}(S \leq Z_\alpha) = 1$, and the histogram of $((LC_n)_{obs} - \gamma_4^* n)/\sqrt{cn}$ is in Figure 3.7.

The experiments show that our proposed testing procedure is effective in that the probability $\mathbb{P}(S \leq Z_\alpha)$ gets closer to zero when the two sequences have higher similarity.

3.3 Upper Bound on the Expected Length of LCSs for Multiple Sequences

For two sequences and equally likely letters from $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, upper bounds on γ_m^* are given in [6], a result which can be extended to an arbitrarily finite number of sequences. Below, following [6], we outline the proof of this extension which will provide upper bounds on $\gamma_{m,d}^*$, where d denotes the number of sequences.

Let $F(n, \boldsymbol{s}, m)$ be the number of sequences of length n that contains $\boldsymbol{s}$, where $\boldsymbol{s}$ is any fixed sequence of length ℓ. Then a counting and inductive argument developed in [6] gives:

Lemma 3.1

$$F(n, \boldsymbol{s}, m) = \sum_{j=\ell}^{n} \binom{n}{j} (m-1)^{n-j}. \tag{3.2}$$

Since

$$\binom{n}{j+1}(m-1)^{n-j-1} \leq \binom{n}{j}(m-1)^{n-j}, \text{ for } j \geq n/m,$$

(3.2) leads to

$$F(n, \boldsymbol{s}, m) \leq n\binom{n}{\ell}(m-1)^{n-\ell}, \text{ for } \ell \geq n/m \tag{3.3}$$

For a fixed $\boldsymbol{s}$ of length ℓ, the number of ordered d-tuples of length-n sequences $(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_d)$ that all contains $\boldsymbol{s}$ as a subsequence is $F^d(n, \boldsymbol{s}, m)$. Then the total number of such $(d+1)$-tuples $(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_d, \boldsymbol{s})$ is

$$G(n, \ell, m) = \sum_{|\boldsymbol{s}|=\ell} F^d(n, \boldsymbol{s}, m),$$

where the summation is over all the m^ℓ sequences of length ℓ.

Now, let $g(n, \ell, m)$ be the number of d-tuples $(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_d)$ such that $LC(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_d) \geq \ell$, then

$$g(n, \ell, m) \leq G(n, \ell, m). \tag{3.4}$$

Next, let $h_m^{(n)}(\theta)$ be the proportion of all ordered $(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_d)$ such that $LC(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_d) \geq \ell$.

Lemma 3.2 *Let $\theta = \ell/n$, then $h_m^{(n)} \leq (H_m(\theta))^{dn}$, where*

$$H_m(\theta) = \frac{m^{(\theta/d)-1}(m-1)^{1-\theta}}{\theta^\theta (1-\theta)^{1-\theta}}.$$

Moreover, $H_m(\theta) = 1$ has a unique solution in the interval $[1/m, 1)$. Let V_m be this solution,

then $H_m(\theta) < 1$, *for* $\theta > V_m$.

Proof. By Lemma 1 as well as (3.3) and (3.4),

$$h_m^{(n)} = \frac{g(n,\ell,m)}{m^{dn}} \le \frac{G(n,\ell,m)}{m^{dn}} = \sum_{|\boldsymbol{s}|=\ell} \frac{F^d(n,\boldsymbol{s},m)}{m^{dn}} \le m^{\ell-dn}\left\{n\binom{n}{\ell}(m-1)^{n-\ell}\right\}^d.$$

Thus by Stirling's formula, ,

$$\begin{aligned}
\lim_{n\to\infty}(h_m^{(n)})^{1/n} &\le \lim_{n\to\infty} m^{(\ell-dn)/n}\left\{n\binom{n}{\ell}(m-1)^{n-\ell}\right\}^{d/n} \\
&= m^{\theta-d}(m-1)^{d-d\theta}\lim_{n\to\infty}\left\{n\binom{n}{\ell}\right\}^{d/n} \\
&= m^{\theta-d}(m-1)^{d-d\theta}\frac{1}{\theta^{d\theta}(1-\theta)^{d-d\theta}} \\
&= H_m(\theta)^d.
\end{aligned}$$

To prove the second statement of the lemma, note that $H_m(\theta) > 0$ for all $\theta \in [1/m, 1)$ and that

$$\lim_{\theta\to 1} H_m(\theta) = m^{1/d-1}\lim_{\theta\to 1}\frac{(m-1)^{1-\theta}}{\theta^\theta(1-\theta)^{1-\theta}} = m^{-(d-1)/d} < 1,$$

while

$$H_m(1/m) = m^{1/dm} > 1.$$

But for $\theta \in [1/m, 1)$,

$$\frac{H_m'(\theta)}{H_m(\theta)} = \log\frac{(1-\theta)m^{1/d}}{(m-1)\theta} = \begin{cases} > 0 & \text{if } \theta > \theta_m \\ < 0 & \text{if } \theta < \theta_m, \end{cases}$$

for some θ_m. Therefore, there exists a unique solution $V_m \in [1/m, 1)$, and $H_m(\theta) < 1$ for $\theta > V_m$. □

Combining the above results leads to:

Proposition 3.3

$$\lim_{n\to\infty} \frac{\mathbb{E}LC_n}{n} \le V_m.$$

Proof. For any $\epsilon > 0$ satisfying $V_m + \epsilon < 1$, separate the total m^{dn} tuples of $(\boldsymbol{a_1}, \boldsymbol{a_2}, \cdots, \boldsymbol{a}_d)$ into two categories: those with longest common subsequences longer than $(V_m + \epsilon)n$, and those with longest common subsequences with length at most $(V_m + \epsilon)n$. Thus,

$$\begin{aligned}
\mathbb{E}LC_n &\le (V_m + \epsilon)n \left\{1 - h_m^{(n)}(V_m + \epsilon)\right\} + (V_m + \epsilon)n \left\{h_m^{(n)}(V_m + \epsilon)\right\} \\
&\le (V_m + \epsilon)n + (V_m + \epsilon)n \left\{h_m^{(n)}(V_m + \epsilon)\right\} \\
&\le (V_m + \epsilon)n + (V_m + \epsilon)nH_m^{dn}(V_m + \epsilon).
\end{aligned}$$

Since $H_m(\theta) < 1$ for $\theta > V_m$, the last term converges to 0 as $n \to \infty$. Thus,

$$\lim_{n\to\infty} \frac{\mathbb{E}LC_n}{n} \le V_m + \epsilon,$$

holds for any ϵ satisfying $V_m + \epsilon < 1$. □

Therefore, from the above proposition, $V_m \in [1/m, 1)$ such that $H_m(V_m) = 1$ provides an upper bound on $\gamma^*_{m,d}$. In particular, letting $m = 2$, i.e., $\mathcal{A} = \{\alpha_1, \alpha_2\}$, leads to Table 3.2 for $\gamma^*_{2,d}$, where the lower bounds are obtained in [18].

Table 3.2: Upper and lower bounds for $\gamma^*_{2,d}$

number of sequences d	upper bound for $\gamma^*_{2,d}$	lower bound for $\gamma^*_{2,d}$
2	0.866595	0.781281
3	0.793026	0.704473
4	0.749082	0.661274
5	0.719527	0.636022
6	0.698053	0.617761
7	0.681605	0.602493
8	0.668516	0.594016
9	0.657797	0.587900
10	0.648819	0.570155

The results of [6] have been improved in [9]. The current multi-sequence result can similarly be improved using the approach there. In particular, this gives for three sequences with binary alphabet, the upper bound 0.791, which is slightly better than 0.793026 obtained above. However for four (or more) sequences, even with an alphabet of size 2, this approach becomes rather cumbersome. Simulation results on $\mathbb{E}LC_n$ are also presented, in some multisequence cases, in [24].

CHAPTER 4

VARIANCE OF THE LENGTH OF THE LONGEST COMMON SUBSEQUENCES IN RANDOM WORDS WITH AN OMITTED LETTER

In this chapter, we study an asymmetrical case where the letters might not be uniformly distributed and where the two sequences may have different distributions of letters. Again, $\boldsymbol{X} = (X_i)_{i\geq 1}$ and $\boldsymbol{Y} = (Y_i)_{i\geq 1}$ are two independent sequences of i.i.d. random variables taking their values in a finite common alphabet $\mathcal{A}$, with $\mathbb{P}(X_1 = \alpha) = p_{x,\alpha} \geq 0$ and $\mathbb{P}(Y_1 = \alpha) = p_{y,\alpha} \geq 0$, $\alpha \in \mathcal{A}$, and let LC_n be the largest k such that there exist $1 \leq i_1 < \cdots < i_k \leq n$ and $1 \leq j_1 < \cdots < j_k \leq n$ with $X_{i_s} = Y_{j_s}$ for $s = 1, \ldots, k$. In words, LC_n is the length of the longest common subsequences of the random words $\boldsymbol{X}^{(n)} := X_1 \cdots X_n$ and $\boldsymbol{Y}^{(n)} := Y_1 \cdots Y_n$.

4.1 Statement of the Results

To specifically state our problem and present our framework, let $\mathcal{A} := \mathcal{A}_{m+1} = \{\alpha_1, \alpha_2, \cdots, \alpha_m, \alpha_{m+1}\}$, and let the letters distribution of $\boldsymbol{X}$ to be such that

$$\mathbb{P}(X_1 = \alpha_1) = \cdots = \mathbb{P}(X_1 = \alpha_m) = \frac{1-p}{m} > 0, \quad \mathbb{P}(X_1 = \alpha_{m+1}) = p > 0,$$

while the letters distribution of $\boldsymbol{Y}$ is such that

$$\mathbb{P}(Y_1 = \alpha_1) = \cdots = \mathbb{P}(Y_1 = \alpha_m) = \frac{1}{m}.$$

To start with, an upper bound on the variance of LC_n is shown to be

$$\operatorname{Var} LC_n \leq \frac{n}{2}\left(2 - p^2 - \frac{1 + (1-p)^2}{m}\right),$$

for all $n \in \mathbb{N}$. Indeed, the Efron-Stein inequality states that:

$$\operatorname{Var} S \leq \frac{1}{2} \sum_{i=1}^{n} \mathbb{E}(S - S_i)^2, \tag{4.1}$$

where, $S = S(Z_1, Z_2, \cdots, Z_n)$ and $S_i = S(Z_1, Z_2, \cdots, Z_{i-1}, \hat{Z}_i, Z_{i+1}, \cdots Z_n)$, and where $(Z_i)_{1 \leq i \leq n}$ and $(\hat{Z}_i)_{1 \leq i \leq n}$ are independent copies of each other.

Now following [26],

$$\begin{aligned}
&\mathbb{E}|LC_n - LC_n(X_1 \cdots X_{i-1} \hat{X}_i X_{i+1} \cdots X_n; Y_1 \cdots Y_n)|^2 \\
&\quad = \mathbb{E}\left(|LC_n - LC_n(X_1 \cdots X_{i-1} \hat{X}_i X_{i+1} \cdots X_n; Y_1 \cdots Y_n)|^2 \mathbf{1}_{X_i \neq \hat{X}_i}\right) \\
&\quad \leq \mathbb{P}(X_i \neq \hat{X}_i) = 1 - \sum_{i=1}^{m+1} (\mathbb{P}(X_1 = \alpha_i))^2 \\
&\quad = 1 - m\left(\frac{1-p}{m}\right)^2 - p^2 \\
&\quad = (1-p)\left(1 - \frac{1}{m} + p\left(1 + \frac{1}{m}\right)\right),
\end{aligned}$$

since when replacing X_i by $\hat{X}_i$, LC_n changes by at most 1 and at least -1. Similarly,

$$\begin{aligned}
\mathbb{E}|LC_n - LC_n(X_1 \cdots X_n; Y_1 \cdots Y_{i-1} \hat{Y}_i Y_{i+1} \cdots Y_n)|^2 &\leq 1 - \sum_{i=1}^{m} (\mathbb{P}(Y_1 = \alpha_i))^2 \\
&= 1 - \frac{1}{m}.
\end{aligned}$$

Applying (4.1) and combining the two bounds above give,

$$\begin{aligned}
\operatorname{Var} LC_n &\leq \frac{1}{2}\left\{(1-p)\left(1 - \frac{1}{m} + p\left(1 + \frac{1}{m}\right)\right)n + \left(1 - \frac{1}{m}\right)n\right\} \\
&= \frac{n}{2}\left(2 - p^2 - \frac{1 + (1-p)^2}{m}\right).
\end{aligned} \tag{4.2}$$

To match the easy bound (4.2), we can now state the main result of this paper.

Theorem 4.1 *There exists a constant $C = C(p, m) > 0$ independent of n, such that for all*

$n \geq 1$,

$$\operatorname{Var} LC_n \geq Cn. \tag{4.3}$$

This theorem, combined with the upper bound (4.2), gives a linear order, in n, for the variance of LC_n, and we refer the reader to Section 4.4 for an estimate on C.

4.2 Proof of Theorem 4.1

In this section, N denotes the number of letters α_{m+1} in the random word $\boldsymbol{X}^{(n)}$. Clearly, N is a binomial random variable with parameter n and p. Moreover, let $\tilde{\boldsymbol{X}}^{(n)} := X_{i_1} \cdots X_{i_k}$, where $1 \leq i_1 < \cdots < i_k \leq n$, $X_j \neq \alpha_{m+1}$ for all $j \in \{i_1, \ldots, i_k\}$ and $X_j = \alpha_{m+1}$ for all $j \in \{1, 2, \ldots, n\} \backslash \{i_1, \ldots, i_k\}$. In words, $\tilde{\boldsymbol{X}}^{(n)}$ is the subword of $\boldsymbol{X}^{(n)}$ made only of non-α_{m+1} letters. To prove our main theorem, we will recursively define a finite random sequence $\boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}, \ldots, \boldsymbol{Z}^{(n)}$, where each $\boldsymbol{Z}^{(k)}$ has length k, by inserting uniformly at random and at a uniform random location a letter from $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ to the previous $\boldsymbol{Z}^{(k-1)}$.

To formally describe the defining mechanism, let $\{U_k\}_{1 \leq k \leq n}$ and $\{T_k\}_{3 \leq k \leq n}$ be two independent sequences of random variables, where $\{U_k\}_{1 \leq k \leq n}$ is a sequence of i.i.d. uniform random variables on $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, and $\{T_k\}_{3 \leq k \leq n}$ is a sequence of independent random variables uniform on $\{2, 3, \ldots, k-1\}$, $k \geq 3$.

Then as in [3], recursively define the sequence $\boldsymbol{Z}^{(k)}$ via:

(1) $\boldsymbol{Z}^{(1)} = U_1$.

(2) $\boldsymbol{Z}^{(2)} = U_1 U_2$.

(3) For $k \geq 2$, given $\boldsymbol{Z}^{(k)} = Z_1^k Z_2^k \cdots Z_k^k$, let $\boldsymbol{Z}^{(k+1)}$ be as follows:

- For all $j < T_{k+1}$, let
$$Z_j^{k+1} = Z_j^k.$$
- For $j = T_{k+1}$, let
$$Z_j^{k+1} = U_{k+1}.$$

- For all j such that $T_{k+1} < j \leq k+1$, let

$$Z_j^{k+1} = Z_{j-1}^k.$$

Hence, $\{Z_i^k\}_{1 \leq i \leq k \leq n}$ is a triangular array of uniform random variables with values in $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, and finding the relation between $\boldsymbol{Z}^{(n-N)}$ and $\tilde{\boldsymbol{X}}^{(n)}$ is the purpose of our next lemma whose proof is akin to a corresponding proof in [15].

Lemma 4.2 *For any $n \geq 1$ and $1 \leq k \leq n$,*

$$\boldsymbol{Z}^{(k)} \stackrel{d}{=} (\tilde{\boldsymbol{X}}^{(n)} | N = n - k),$$

and moreover,

$$\boldsymbol{Z}^{(n-N)} \stackrel{d}{=} \tilde{\boldsymbol{X}}^{(n)},$$

where $\stackrel{d}{=}$ denotes equality in distribution.

Proof. The proof is by induction on k. Let $k = 1$, by definition, $\boldsymbol{Z}^{(1)} = U_1$, which has the same distribution as $(\tilde{\boldsymbol{X}}^{(n)} | N = n - 1)$. Next, assume that

$$\boldsymbol{Z}^{(k)} \stackrel{\mathrm{d}}{=} (\tilde{\boldsymbol{X}}^{(n)} | N = n - k),\ 2 \leq k \leq n - 1,$$

and so for any $(\alpha_{j_1}, \alpha_{j_2}, \ldots, \alpha_{j_k}) \in \mathcal{A}^k$,

$$\mathbb{P}\left((Z_1^k, Z_2^k, \ldots, Z_k^k) = (\alpha_{j_1}, \alpha_{j_2}, \ldots, \alpha_{j_k})\right) = \left(\frac{1}{m}\right)^k.$$

Then,

$$
\begin{aligned}
&\mathbb{P}\left((Z_1^{k+1}, Z_2^{k+1}, \ldots, Z_{k+1}^{k+1}) = (\alpha_{j_1'}, \alpha_{j_2'}, \ldots, \alpha_{j_{k+1}'})\right) \\
&\quad = \sum_{t=2}^{k} \mathbb{P}\left((Z_1^{k+1}, Z_2^{k+1}, \ldots, Z_{k+1}^{k+1}) = (\alpha_{j_1'}, \alpha_{j_2'}, \ldots, \alpha_{j_{k+1}'}) | T_{k+1} = t\right) \mathbb{P}(T_{k+1} = t) \\
&\quad = \sum_{t=2}^{k} \mathbb{P}\left((Z_1^{k}, \ldots, Z_{t-1}^{k}, Z_t^{k}, \ldots, Z_k^{k}) = (\alpha_{j_1'}, \ldots, \alpha_{j_{t-1}'}, \alpha_{j_{t+1}'}, \ldots, \alpha_{j_{k+1}'})\right) \mathbb{P}(U_{k+1} = \alpha_{j_t'}) \mathbb{P}(T_{k+1} = t) \\
&\quad = \sum_{t=2}^{k} \left(\frac{1}{m}\right)^k \frac{1}{m} \frac{1}{k-1} \\
&\quad = \left(\frac{1}{m}\right)^{k+1}.
\end{aligned}
$$

Thus,

$$\boldsymbol{Z}^{(k+1)} \overset{\mathrm{d}}{=} (\tilde{\boldsymbol{X}}^{(n)} | N = n - k - 1).$$

To prove the second part of the lemma, from the independence of N and $\boldsymbol{Z}^{(n-k)}$, for any $u \in \mathbb{R}^{n-k}$,

$$
\begin{aligned}
\mathbb{E}e^{i<u, \tilde{\boldsymbol{X}}^{(n)}>} &= \sum_{k=0}^{n} \mathbb{E}\left(e^{i<u, \tilde{\boldsymbol{X}}^{(n)}>} | N = k\right) \mathbb{P}(N = k) \\
&= \sum_{k=0}^{n} \mathbb{E}\left(e^{i<u, \boldsymbol{Z}^{(n-k)}>}\right) \mathbb{P}(N = k) \\
&= \sum_{k=0}^{n} \mathbb{E}\left(e^{i<u, \boldsymbol{Z}^{(n-k)}>} | N = k\right) \mathbb{P}(N = k) \\
&= \sum_{k=0}^{n} \mathbb{E}\left(e^{i<u, \boldsymbol{Z}^{(n-N)}>} | N = k\right) \mathbb{P}(N = k) \\
&= \mathbb{E}e^{i<u, \boldsymbol{Z}^{(n-N)}>}.
\end{aligned}
$$

Thus,

$$\boldsymbol{Z}^{(n-N)} \overset{\mathrm{d}}{=} \tilde{\boldsymbol{X}}^{(n)}.$$

□

Now let LC_n be the length of the longest common subsequences of $\boldsymbol{X}^{(n)}$ and $\boldsymbol{Y}^{(n)}$, and let $L_n(k)$ be the length of the longest common subsequences/subwords of $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{Y}^{(n)}$. It follows from Lemma 4.2 that,

$$LC_n \overset{\mathrm{d}}{=} L_n(n-N), \tag{4.4}$$

and therefore,

$$\operatorname{Var} LC_n = \operatorname{Var}(L_n(n-N)). \tag{4.5}$$

In order to prove the main result, we will also need the following result taken from [15].

Lemma 4.3 *Let $f : D \subset \mathbb{R} \to \mathbb{Z}$ satisfy a local reversed Lipschitz condition, i.e., let $h \geq 0$ and let f be such that for any $i, j \in D$ with $j \geq i + h$,*

$$f(j) - f(i) \geq c(j-i),$$

for some $c > 0$. Let T be a D-valued random variable with $\mathbb{E}|f(T)|^2 < \infty$, then

$$\operatorname{Var} f(T) \geq \frac{c^2}{2}\left(\operatorname{Var}(T) - h^2\right).$$

Next, let

$$O_n := \bigcap_{\substack{i,j\in I \\ j\geq i+h(n)}} \{L_n(j) - L_n(i) \geq K(j-i)\}, \tag{4.6}$$

where $I = [np - \sqrt{np(1-p)}, np + \sqrt{np(1-p)}]$, $K > 0$ is a constant which does not depend on n ($K \leq 1/2m$ will do, see Lemma 4.13), and where $h(n)$ will also be made precise later. The event O_n can be viewed as the event where the map $k \to L_n(k)$ locally satisfies a reversed Lipschitz condition.

In Section 4.3, we will prove

Theorem 4.4 *For all $n \geq 1$,*

$$\mathbb{P}(O_n) \geq 1 - Ae^{-Bn} - ne^{-2K^2 h(n)}, \tag{4.7}$$

where, K is given in Lemma 4.13, $A = \max\{C_4, C_5, C_7\}$, and $B = \min\{C_3\nu, C_6, C_8\}$, and these constants are given in (4.22)*, Lemma 4.9, and Lemma 4.11 respectively.*

Now with the help of Theorem 4.4 we can provide the proof of our main result stated in Theorem 4.1.

Proof of Theorem 4.1. By (4.5), it is sufficient to prove the lower bound for $\mathrm{Var}(L_n(n-N))$. First as in [15], with its notation,

$$\begin{aligned}\mathrm{Var}(U|V) &\le 2^2\left(\mathbb{E}((U-\mathbb{E}U)^2\,\big|\,V)/2 + \mathbb{E}((\mathbb{E}(U|V)-\mathbb{E}U)^2\,\big|\,V)/2\right)\\ &\le 2^2\mathbb{E}((U-\mathbb{E}U)^2\,\big|\,V),\end{aligned} \tag{4.8}$$

and so, for any $n \ge 1$,

$$\begin{aligned}\mathrm{Var}(L_n(n-N)) &\ge \frac{1}{2^2}\mathbb{E}\left(\mathrm{Var}(L_n(n-N)\,|\,(L_n(n-k))_{0\le k\le n})\right)\\ &= \frac{1}{2^2}\int_\Omega \mathrm{Var}(L_n(n-N)\,|\,(L_n(n-k))_{0\le k\le n}(\omega))\mathbb{P}(d\omega)\\ &\ge \frac{1}{2^2}\int_{O_n} \mathrm{Var}(L_n(n-N)\,|\,(L_n(n-k))_{0\le k\le n}(\omega))\mathbb{P}(d\omega).\end{aligned} \tag{4.9}$$

Since N is independent of $(L_n(n-k))_{0\le k\le n}$, and from (4.8), for each $\omega \in \Omega$,

$$\begin{aligned}&\mathrm{Var}(L_n(n-N)|(L_n(n-k))_{0\le k\le n}(\omega))\\ &\ge \mathrm{Var}(L_n(n-N)|(L_n(n-k))_{0\le k\le n}(\omega), \mathbf{1}_{N\in I}=1)\mathbb{P}(N\in I|(L_n(n-k))_{0\le k\le n}(\omega))\\ &= \mathrm{Var}(L_n(n-N)|(L_n(n-k))_{0\le k\le n}(\omega), \mathbf{1}_{N\in I}=1)\mathbb{P}(N\in I),\end{aligned} \tag{4.10}$$

where again,

$$I = \left[np - \sqrt{n(1-p)p}, np + \sqrt{n(1-p)p}\right].$$

Again, for each $\omega \in O_n$, from Lemma 4.3, and since N is independent of $(L_n(n-k))_{0\le k\le n}$,

$$\mathrm{Var}(L_n(n-N)|(L_n(n-k))_{0\le k\le n}(\omega), \mathbf{1}_{N\in I}=1) \ge \frac{K^2}{8}\left(\mathrm{Var}(N|\mathbf{1}_{N\in I}=1) - h(n)^2\right). \quad (4.11)$$

Now, (4.9), (4.10) and (4.11) give

$$\mathrm{Var}(L_n(n-N)) \ge \frac{K^2}{8}\left(\mathrm{Var}(N|\mathbf{1}_{N\in I}=1) - h(n)^2\right)\mathbb{P}(N\in I)\mathbb{P}(O_n), \quad (4.12)$$

and it remains to estimate each one of the three terms on the right hand side of (4.12). By the Berry-Esséen inequality, and all $n \ge 1$,

$$\left|\mathbb{P}(N\in I) - \frac{1}{\sqrt{2\pi}}\int_{-1}^{1} e^{-\frac{x^2}{2}}dx\right| \le \frac{1}{\sqrt{np(1-p)}}. \quad (4.13)$$

Moreover,

$$\begin{aligned}
&\mathrm{Var}(N|\mathbf{1}_{N\in I}=1)\\
&= \mathbb{E}((N-np+np-\mathbb{E}(N|\mathbf{1}_{N\in I}=1))^2\,|\mathbf{1}_{N\in I}=1)\\
&\ge \left(\mathbb{E}((N-np)^2|\mathbf{1}_{N\in I}=1)^{1/2} - |np-\mathbb{E}(N|\mathbf{1}_{N\in I}=1)|\right)^2, \quad (4.14)
\end{aligned}$$

and

$$\begin{aligned}
&|\mathbb{E}(N|\mathbf{1}_{N\in I}=1)-np|\\
&= \sqrt{np(1-p)}\left|\mathbb{E}\left(\frac{N-np}{\sqrt{np(1-p)}}\Big|\mathbf{1}_{N\in I}=1\right)\right|\\
&= \sqrt{np(1-p)}\frac{\left|F_n(1)-\Phi(1)+F_n(-1)-\Phi(-1)-\int_{-1}^{1}(F_n(x)-\Phi(x))dx\right|}{\mathbb{P}(N_1\in I)}\\
&\le \sqrt{np(1-p)}\frac{4\max_{x\in[-1,1]}|F_n(x)-\Phi(x)|}{\mathbb{P}(N_1\in I)}\\
&\le \frac{2}{\int_{-1}^{1} e^{-\frac{x^2}{2}}dx/\sqrt{2\pi} - 1/\sqrt{np(1-p)}}, \quad (4.15)
\end{aligned}$$

where F_n is the distribution functions of $(N-np)/\sqrt{np(1-p)}$, while Φ is the standard normal one. Likewise,

$$\begin{aligned}
&\mathbb{E}(|N-np|^2|\mathbf{1}_{N\in I}=1)\\
&\qquad\geq (np(1-p))\frac{\int_{-1}^{1}|x|^2 d\Phi(x)-4\max_{x\in[-1,1]}|F_n(x)-\Phi(x)|}{\mathbb{P}(N_1\in I)}\\
&\qquad\geq (np(1-p))\frac{\int_{-1}^{1}|x|^2 e^{-\frac{x^2}{2}}dx-2\sqrt{2\pi}/\sqrt{np(1-p)}}{\int_{-1}^{1}e^{-\frac{x^2}{2}}dx+\sqrt{2\pi}/\sqrt{np(1-p)}}. \qquad (4.16)
\end{aligned}$$

Next, using (4.14) – (4.16),

$$\begin{aligned}
&\mathrm{Var}(N|\mathbf{1}_{N\in I}=1)\\
&\geq \left|(np(1-p))^{\frac{1}{2}}\left(\frac{\int_{-1}^{1}|x|^2 e^{-\frac{x^2}{2}}dx-2\sqrt{2\pi}/\sqrt{np(1-p)}}{\int_{-1}^{1}e^{-\frac{x^2}{2}}dx+\sqrt{2\pi}/\sqrt{np(1-p)}}\right)^{\frac{1}{2}}\right.\\
&\qquad\left.-\frac{2}{\int_{-1}^{1}e^{-\frac{x^2}{2}}dx/\sqrt{2\pi}-1/\sqrt{np(1-p)}}\right|^2. \qquad (4.17)
\end{aligned}$$

Finally, the estimates (4.12)-(4.17) combined with the estimate on $\mathbb{P}(O_n)$ obtained in Theorem 4.4 give the lower bound in Theorem 4.1, whenever $2\ln n/K^2 \leq h(n) \leq K_1\sqrt{n}$, where the upper bound on $h(n)$ stems from the requirement that the right hand side of (4.12) needs to be lower bounded and where K_1 is estimated in Section 4.4.

□

4.3 Proof of Theorem 4.4

In this section, we prove the aforementioned theorem, therefore completing our proof of Theorem 4.1. Before doing so, we will need to state a few definitions and set some notations used throughout the rest of the paper:

The sequences $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{Y}^{(n)}$ are said to have a common subsequence of length ℓ if there

exist increasing functions $\pi : [1, \ell] \to [1, k]$ and $\eta : [1, \ell] \to [1, n]$ such that

$$Z^k_{\pi(i)} = Y_{\eta(i)}, \quad i = 1, 2, \ldots, \ell,$$

and (π, η) is then called a pair of matching subsequences of $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{Y}^{(n)}$. Also, throughout, M^k denotes the set of pairs of matching subsequences of $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{Y}^{(n)}$ of maximal length.

The proof of Theorem 4.4 is then divided into two cases, $k < \nu n$ and $k \geq \nu n$, where in each case $\nu < 1/m$.

4.3.1 $k < \nu n$ ($\nu < 1/m$)

We begin with the simpler case $k < \nu n$. In this situation, we show that with high probability all the letters of $\boldsymbol{Z}^{(k)}$ are matched with letters of $\boldsymbol{Y}^{(n)}$. Let

$$E^{(n)}_k := \{L_n(k) = k\}.$$

Then clearly, $E^{(n)}_k \subset E^{(n)}_{k-1} \subset \cdots \subset E^{(n)}_1$, and so

$$E^{(n)} := \bigcap_{k=1}^{\nu n} E^{(n)}_k = E^{(n)}_{\nu n} = \{L_n(k+1) - L_n(k) = 1, \forall\ k < \nu n\}.$$

Lemma 4.5 *For $\nu < 1/m$, there exists a constant $C_1 = C_1(\nu, m) > 0$ such that,*

$$\mathbb{P}(L_n(\nu n) = \nu n) \geq 1 - \exp(-C_1 n).$$

Proof. We construct a pair of matching sequence (π, η) for $\boldsymbol{Z}^{(k)} = Z^k_1 Z^k_2 \cdots Z^k_k$ and $\boldsymbol{Y}$ as follows,

$$\begin{cases} \pi(i) = i, \\ \eta(i) = \min\{\ell :\ \ell > \eta(i-1), Y_\ell = Z^k_i\}, \end{cases} \qquad \text{for } i \geq 1,$$

where we also set $\eta(0) = 0$.

Thus, $\eta(i)$ is the smallest index ℓ such that $Z_1^k \cdots Z_i^k$ is a subsequence of $Y_1 Y_2 \cdots Y_\ell$. In this way, $\eta(1), \eta(2), \eta(3), \cdots$ is a renewal process with geometrically distributed holding time, i.e., denoting the inter arrival times as

$$T_i = \eta(i) - \eta(i-1),$$

then $\{T_i\}_{i \geq 1}$ is a sequence of independent geometric random variables with parameter $1/m$, i.e.,

$$\mathbb{P}(T_i = t) = \frac{1}{m}\left(\frac{m-1}{m}\right)^{t-1}, \ t = 1, 2, 3, \cdots .$$

Thus, $\mathbb{E}T_i = m$. Next,

$$\mathbb{P}(L_n(\nu n) = \nu n) \geq \mathbb{P}\left(\sum_{i=1}^{\nu n} T_i < n\right) = 1 - \mathbb{P}\left(\sum_{i=1}^{\nu n}\left(T_i - \frac{1}{\nu}\right) \geq 0\right),$$

and from the independence of the $\{T_i\}_{i \geq 1}$,

$$\begin{aligned}
\mathbb{P}\left(\sum_{i=1}^{\nu n}\left(T_i - \frac{1}{\nu}\right) \geq 0\right) &\leq \inf_{s>0} \mathbb{E}\left(e^{s\sum_{i=1}^{\nu n}\left(T_i - \frac{1}{\nu}\right)}\right) \\
&= \inf_{s>0}\left(\mathbb{E}e^{s(T_0 - 1/\nu)}\right)^{\nu n} \\
&= \inf_{s>0} e^{-ns}\left(\frac{e^s}{m-(m-1)e^s}\right)^{\nu n}.
\end{aligned}$$

This last term is minimized at

$$s = \ln \frac{m(1-\nu)}{m-1},$$

thus,

$$\inf_{s>0} e^{-s}\left(\frac{e^s}{m-(m-1)e^s}\right)^{\nu} = \frac{(1-\nu)^{\nu-1}}{m(m-1)^{\nu-1}\nu^{\nu}},$$

which is increasing in ν for $\nu \in (0, 1 - 1/m)$. Thus,

$$\frac{(1-\nu)^{\nu-1}}{m(m-1)^{\nu-1}\nu^{\nu}} \begin{cases} > 1 & \text{when } \nu \in (1/m, 1-1/m) \\ = 1 & \text{when } \nu = 1/m \\ < 1 & \text{when } \nu < 1/m. \end{cases}$$

Since $\nu < 1/m$, by taking $C_1 = \ln\left(m(m-1)^{\nu-1}\nu^{\nu}/(1-\nu)^{\nu-1}\right)$, we have

$$\mathbb{P}(E_{\nu n}^{(n)}) = \mathbb{P}(L_n(\nu n) = \nu n) \geq 1 - \exp(-C_1 n).$$

□

Therefore, Lemma 4.5 asserts that

$$\mathbb{P}(E^{(n)}) = \mathbb{P}(E_{\nu n}^{(n)}) \geq 1 - \exp(-C_1 n).$$

4.3.2 $k \geq \nu n$ ($\nu < 1/m$)

To continue, we introduce some more definitions and notations of use throughout the section.

(i) Let $\leq$ denote the partial order between two increasing functions $\pi_1, \pi_2 : [1, \ell] \to \mathbb{N}$, i.e., $\pi_1 \leq \pi_2$ if for every $i \in [1, \ell]$, $\pi_1(i) \leq \pi_2(i)$. Further $(\pi_1, \eta_1) \leq (\pi_2, \eta_2)$ is short for $\pi_1 \leq \pi_2$ and $\eta_1 \leq \eta_2$.

(ii) Let $M_{min}^k \subset M^k$ be the set of $(\pi, \eta) \in M^k$ which are minimal for the relation $\leq$, i.e., such that for $(\pi_1, \eta_1) \in M_{min}^k$ and $(\pi_2, \eta_2) \in M^k$, if $(\pi_1, \eta_1) \geq (\pi_2, \eta_2)$ then $(\pi_1, \eta_1) = (\pi_2, \eta_2)$.

(iii) If (π, η) is a pair of matching subsequences of $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{Y}^{(n)}$ of length ℓ, a match of (π, η) is then defined to be the quadruple

$$(\pi(i), \pi(i+1), \eta(i), \eta(i+1)).$$

Moreover, if $\eta(i)+2 \le \eta(i+1)$, the match is said to be non-empty. Therefore, for a non-empty match, there exists j, such that $\eta(i) < j < \eta(i+1)$ and $Y_j = \alpha$ for some $\alpha \in \mathcal{A} \setminus \{\alpha_{m+1}\}$. In that case, the match is said to contain an α, and Y_j is called an unmatched letter of the match $(\pi(i), \pi(i+1), \eta(i), \eta(i+1))$.

(iv) The sequence $\boldsymbol{Y}^{(n)}$ can be uniquely divided into d compartments $[j_1, j_2-1], [j_2, j_3-1], \ldots, [j_d, n]$, where $1 = j_1 < j_2 < \cdots < j_d \le n$ are determined by the following recursive relations:

$$\begin{cases} j_1 = 1 \\ j_i = \min(n+1, \{s \in [j_{i-1}+1, n] : \ Y_{j_{i-1}} Y_{j_{i-1}+1} \cdots Y_s \text{ contains } m \text{ distinct letters}\}), \end{cases},$$

and $d = \max\{i : \ j_i \le n\}$.

To get a lower bound on the probability that the length of the longest common subsequence increases by one, we recall the construction of $\boldsymbol{Z}^{(k)}$ and note that there are $(k-1)$ possible positions for the letter U_{k+1} to be inserted. Therefore, U_{k+1} falls into a non-empty match with probability at least (number of nonempty matches of (π, η))$/(k-1) \ge$ (number of nonempty matches of (π, η))$/k$. For each non-empty match, there is at least one unmatched letter, and the probability that U_{k+1} takes the same value as the unmatched letter is $1/m$, resulting in the following lower bound for $(\pi, \eta) \in M^k$:

$$\mathbb{P}\left(L_n(k+1) - L_n(k) = 1 | \boldsymbol{Z}^{(k)}, \boldsymbol{Y}^{(n)}\right) \ge \frac{1}{m} \frac{\text{number of nonempty matches of } (\pi, \eta)}{k}. \quad (4.18)$$

Therefore, a good estimate on the number of nonempty matches of (π, η) will provide a lower bound on the probability that LC_n increases by one.

Next we give the main ideas behind the proof that, with high probability, the map $k \to L(k)$ is linearly increasing on $[\nu n, n]$. We use the letter-insertion scheme, described above, to prove that the random map $k \to L(k)$ typically has a positive drift λ (which will

be determined later in Lemma 4.12). To do so, let

$$F_k^{(n)} = \{(\pi, \eta) \in M_{min}^k \text{ such that the number of nonempty matches of } (\pi, \eta) \text{ is at least } \lambda n\}. \tag{4.19}$$

When $F_k^{(n)}$ holds, every pair of $(\pi, \eta) \in M_{min}^k$ has at least λn nonempty matches. Hence the number of non-empty matches divided by k is larger than or equal to $\lambda n/k$. It follows from (4.18) that when $F_k^{(n)}$ holds,

$$\mathbb{P}(L_n(k+1) - L_n(k) = 1 | \boldsymbol{Z}^{(k)}, \boldsymbol{Y}^{(n)}) \geq \frac{1}{m}\frac{\lambda n}{k} \geq \frac{\lambda}{m} > 0. \tag{4.20}$$

Let $F^{(n)}$ be the event

$$F^{(n)} = \bigcap_{k=\nu n}^{n} F_k^{(n)}.$$

The inequality (4.20) implies that when $F^{(n)}$ holds, the map $k \to L_n(k)$ has drift at least λ/m for $k \in [\nu n, n]$. Whenever $F^{(n)}$ holds, with high probability $k \to L_n(k)$ has positive slope on $[\nu n, n]$.

It remains to show that, by concentration, $F^{(n)}$ holds with high probability, and this is proved by contradiction. Indeed if all the matches of $(\pi, \eta) \in M^k$ were empty, then the following two conditions would hold:

(1) $(\eta(1), \eta(2), \eta(3), \cdots, \eta(\ell)) = (\eta(1), \eta(1)+1, \eta(1)+2, \cdots, \eta(1)+\ell-1)$ where ℓ is the length of the LCS of $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{Y}^{(n)}$, i.e., $\ell = L_n(k)$.

(2) The sequence

$$Y_{\eta(1)} Y_{\eta(2)} \cdots Y_{\eta(\ell)} = Y_{\eta(1)} Y_{\eta(1)+1} \cdots Y_{\eta(1)+\ell-1}$$

would be a subsequence of

$$Z_{\pi(1)}^k Z_{\pi(1)+1}^k \cdots Z_{\pi(\ell)}^k.$$

Above, we have two independent sequences of i.i.d. uniform random variables with

parameter $1/m$, where one is contained in the other as a subsequence. Thus, the longer one must approximately be at least m times as long as the shorter one, hence k is approximately at least m times as long as $\ell = L_n(k)$. As a result, the ratio $L_n(k)/k$ is to be at most $1/m$, which is very unlikely (Lemma 4.9), leading to contradiction.

From the previous arguments, it follows that with high probability any $(\pi, \eta) \in M_{min}^k$ contains a non-vanishing proportion $\epsilon > 0$ of unmatched letters, hence $(\eta(L_n(k)) - L_n(k))/\eta(L_n(k)) \geq \epsilon$, where $\eta(L_n(k))$ is the index of the last matching letter in $\boldsymbol{Y}^{(n)}$ of the match (π, η). We then show that this proportion ϵ of unmatched letters generates sufficiently many non-empty matches, i.e., that the unmatched letters should not be concentrated on a too small number of matches.

To prove that there are more than λn nonempty matches, the following two arguments are used:

(1) Any $(\pi, \eta) \in M_{min}^k$ is such that every match of (π, η) contains unmatched letters from at most one compartment of $\boldsymbol{Y}^{(n)}$.

(2) There exists a $D > 0$, not depending on n, such that, with high probability, the total number of integer points contained in the compartments of $\boldsymbol{Y}^{(n)}$ of length larger than D, is small.

Henceforth, for $(\pi, \eta) \in M_{min}^k$ the majority of unmatched letters are at most D per match, ensuring that a proportion ϵ of unmatched letters implies a proportion of at least ϵ/D non-empty matches.

Let us return to the proof, and let $L_\ell(k)$ denote the length of the LCS of $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{Y}^{(\ell)} = Y_1 \cdots Y_\ell$. In order for $\boldsymbol{Y}^{(\ell)}$ to be contained in $\boldsymbol{Z}^{(k)}$, k needs to be approximately m times as long as ℓ, and, then, $L_\ell(k) = \ell$. Therefore, if $k = m\ell(1 - \delta)$, for some $\delta = \delta(\epsilon) > 0$ not depending on ℓ, then it is extremely unlikely that $\boldsymbol{Y}^{(\ell)}$ is a subsequence of $\boldsymbol{Z}^{(k)}$, as shown in the forthcoming lemma.

Lemma 4.6 *For any* $0 < \delta < (m-1)/m$ *and* $\ell \geq 1$*, we have*

$$\mathbb{P}(L_\ell(m\ell(1-\delta)) = \ell) \leq e^{-C_2\delta^2\ell}, \tag{4.21}$$

where $C_2 = m/2(m-1)$.

Proof. The proof is similar to the proof of Lemma 4.5 and some of its notation is used.

First let $\tilde{\boldsymbol{X}} := \tilde{\boldsymbol{X}}^{(\infty)}$, be the (infinite) subword of $\boldsymbol{X}$ with α_{m+1} removed, and therefore each $\tilde{\boldsymbol{X}}^{(n)}$ is a subword of $\tilde{\boldsymbol{X}}$. Next, construct a pair of matching sequence (π, η) for $\tilde{\boldsymbol{X}}$ and $\boldsymbol{Y}^{(\ell)}$ as follows:

$$\pi(0) = 0, \qquad \text{and for } i \geq 1, \quad \begin{cases} \pi(i) = \min\{j : j > \pi(i-1), \tilde{X}_j = Y_i\} \\ \eta(i) = i. \end{cases}$$

Thus, $\pi(i)$ is the smallest index j such that $Y_1Y_2\cdots Y_i$ is a subsequence of $\tilde{X}_1\cdots\tilde{X}_j$. In this way, $\pi(1), \pi(2), \pi(3), \cdots$ is a renewal process with geometrically distributed holding time, i.e., denoting the interarrival times as

$$T_i = \pi(i) - \pi(i-1),$$

then $\{T_i\}_{i\geq 1}$ is a sequence of independent geometric random variables with parameter $1/m$, i.e.,

$$\mathbb{P}(T_i = t) = \frac{1}{m}\left(\frac{m-1}{m}\right)^{t-1}, \; t = 1, 2, 3, \cdots.$$

Thus, $\mathbb{E}T_i = m$. Then by Lemma 4.2 and for $0 < \delta < 1$, we have

$$\begin{aligned} \mathbb{P}(L_\ell(m\ell(1-\delta)) = \ell) &= \mathbb{P}\left(\sum_{i=1}^{\ell} T_i \leq m\ell(1-\delta)\right) = \mathbb{P}\left(\sum_{i=1}^{\ell}(m(1-\delta) - T_i) \geq 0\right) \\ &\leq \left(\inf_{s>0} \mathbb{E}e^{s(m(1-\delta)-T_1)}\right)^{\ell} = \left(\inf_{s>0} \frac{e^{sm(1-\delta)}}{me^s - (m-1)}\right)^{\ell}. \end{aligned}$$

This last term is minimized at

$$s = \ln\left(1 + \frac{\delta}{m(1-\delta)-1}\right),$$

thus setting,

$$w := \inf_{s>0} \frac{e^{sm(1-\delta)}}{me^s - (m-1)} = \frac{(m(1-\delta)-1)\left(1 + \frac{\delta}{m(1-\delta)-1}\right)^{m(1-\delta)}}{m-1},$$

it follows that,

$$\mathbb{P}\left(\sum_{i=1}^{\ell}(m(1-\delta) - T_i) \geq 0\right) \leq e^{(\ln w)\ell}.$$

Now, the Taylor expansion of $\ln w$ with Lagrange remainder gives

$$\ln w = -\frac{m}{2(m-1)}\delta^2 + \frac{1}{6}\left(\frac{m}{(1-\xi)^2} - \frac{m^3}{(m(1-\xi)-1)^2}\right)\delta^3 < -\frac{m}{2(m-1)}\delta^2,$$

where $0 < \xi < \delta$. Letting $C_2 = m/2(m-1)$ finishes the proof of the lemma. □

Lemma 4.6 further entails, as shown next, that for any $0 < \epsilon < 1$ there exists $\delta(\epsilon) > 0$, small, such that $L_\ell(m\ell(1-\delta(\epsilon))) \geq \ell(1-\epsilon)$ is also very unlikely.

Lemma 4.7 *For any $0 < \epsilon < 1$ and all $\ell \geq 1$, there exists $\delta(\epsilon) > 0$, with $\lim_{\epsilon\to 0}\delta(\epsilon) \to 0$, such that*

$$\mathbb{P}\big((G_\ell^{(n)}(\epsilon))^c\big) \leq e^{-C_3\ell},$$

where $G_\ell^{(n)}(\epsilon) = \{L_\ell(m\ell(1-\delta(\epsilon))) < \ell(1-\epsilon)\}$, and where $C_3 := (\delta(\epsilon)-\epsilon)^2 C_2/2$. Therefore, letting

$$G^{(n)}(\epsilon) := \bigcap_{\ell=\nu n}^{n} G_\ell^{(n)}(\epsilon),$$

it follows that,

$$\mathbb{P}(G^{(n)}(\epsilon)) \geq 1 - \sum_{k=\nu n}^{n} e^{-C_3 k} \geq 1 - \frac{1}{1-e^{-C_3}} e^{-C_3\nu n} = 1 - C_4 e^{-C_3\nu n}, \tag{4.22}$$

where $C_4 = 1/(1 - e^{-C_3})$.

Proof. Let $S \subset \{1, 2, \cdots, \ell\}$ have cardinality $(1 - \epsilon)\ell$. Clearly, there are $\binom{\ell}{\ell(1-\epsilon)}$ such subsets S. Now fixing the values of $\boldsymbol{Y}^{(n)}$ at the indices belonging to S, there are $m^{\epsilon\ell}$ such $\boldsymbol{Y}^{(n)}$ agreeing on S. Therefore,

$$\mathbb{P}\big((G_\ell^{(n)}(\epsilon))^c\big) \leq m^{\epsilon\ell}\binom{\ell}{\ell(1-\epsilon)}\mathbb{P}\big(L_{\ell(1-\epsilon)}\left(m\ell\left(1-\delta(\epsilon)\right)\right) = \ell(1-\epsilon)\big)\,.$$

From (4.21),

$$\begin{aligned}\mathbb{P}\big(L_{\ell(1-\epsilon)}\left(m\ell\left(1-\delta(\epsilon)\right)\right) = \ell(1-\epsilon)\big) &= \mathbb{P}\left(L_{\ell(1-\epsilon)}\left(m\left(1-\frac{\delta(\epsilon)-\epsilon}{1-\epsilon}\right)(1-\epsilon)\ell\right) = \ell(1-\epsilon)\right)\\ &\leq e^{-C_2\left(\frac{\delta(\epsilon)-\epsilon}{1-\epsilon}\right)^2(1-\epsilon)\ell} \leq e^{-C_2(\delta(\epsilon)-\epsilon)^2\ell}.\end{aligned}$$

Collecting the above estimates,

$$\mathbb{P}\big((G_\ell^{(n)}(\epsilon))^c\big) \leq m^{\epsilon\ell}\binom{\ell}{\ell(1-\epsilon)}e^{-C_2(\delta(\epsilon)-\epsilon)^2\ell}. \tag{4.23}$$

Since

$$\ell^\ell = (\epsilon\ell + (1-\epsilon)\ell)^\ell \geq \binom{\ell}{(1-\epsilon)\ell}(\epsilon\ell)^{\epsilon\ell}((1-\epsilon)\ell)^{(1-\epsilon)\ell},$$

then

$$\binom{\ell}{(1-\epsilon)\ell} \leq \left(\frac{1}{\epsilon^\epsilon(1-\epsilon)^{1-\epsilon}}\right)^\ell.$$

Therefore, (4.23) becomes

$$\mathbb{P}\big((G_\ell^{(n)}(\epsilon))^c\big) \leq e^{\left(\epsilon(\ln m - \ln\epsilon) - (1-\epsilon)\ln(1-\epsilon) - C_2(\delta(\epsilon)-\epsilon)^2\right)\ell},$$

and it is enough to choose

$$\delta(\epsilon) = \epsilon + \sqrt{\frac{2}{C_2}\left(\epsilon(\ln m - \ln\epsilon) - (1-\epsilon)\ln(1-\epsilon)\right)}, \tag{4.24}$$

to obtain the lemma. □

Lemma 4.8, Lemma 4.9 and Lemma 4.10 presented next, formalize our contradictory argument asserted above. To show that it is unlikely that "the ratio $L_n(k)/k$ is at most $1/m$", we start with:

Lemma 4.8 *For $n \geq 2$, $\mathbb{E}\, L_n(n)/n > 1/m$.*

Proof. For $n \geq 2$,

$$\mathbb{E}L_n(n) > \mathbb{E}\sum_{i=1}^{n} \mathbb{1}_{\{Y_i = Z_i^n\}} \geq n\mathbb{P}(Y_1 = Z_1^n) = \frac{n}{m}.$$

Therefore, $\mathbb{E}\, L_n(n)/n > 1/m$. □

Specifically, when $n = 2$, see [5],

$$\mathbb{E}\, L_n(n)/n = \mathbb{E}\, L_2(2)/2 = \frac{4m^2 - 5m + 3}{2m^3}.$$

Now let ξ_m be such that

$$1/m < \xi_m < \mathbb{E}\, L_2(2)/2, \tag{4.25}$$

and let us show that very likely $L_n(k)/k$ is larger than ξ_m. To do so, let

$$H_k^{(n)} := \{L_n(k) \geq \xi_m k\}$$

and

$$H^{(n)} := \bigcap_{k=\nu n}^{n} H_k^{(n)}.$$

Lemma 4.9 *There exist constants $C_5, C_6 > 0$, such that*

$$\mathbb{P}(H^{(n)}) \geq 1 - C_5 e^{-C_6 n}. \tag{4.26}$$

Proof. Divide the sequences $\boldsymbol{Z}^{(k)}$ and $\boldsymbol{Y}^{(n)}$ into subsequences of length 2, as given in the previous lemma. Then, by superadditivity, $L_k(k) \geq \sum_{i=1}^{k/2} \hat{L}_i$, where $\hat{L}_i$ is the length of the longest common subsequence between $Y_{2(i-1)+1}Y_{2i}$ and $Z^k_{2(i-1)+1}Z^k_{2i}$. Clearly, by the i.i.d. assumptions, $\mathbb{E}(\hat{L}_i) = \mathbb{E}(L_2(2))$ is constant. Hence for $\tau > 0$,

$$\mathbb{P}\left(\sum_{i=1}^{k/2} \hat{L}_i < k\left(\frac{\mathbb{E}(\hat{L}_i) - \tau}{2}\right)\right) \leq \left(\inf_{s<0} \mathbb{E}\left(e^{s\left(\hat{L}_1 - (\mathbb{E}(L_2(2)) - \tau)\right)}\right)\right)^{\frac{k}{2}}. \tag{4.27}$$

Now let $p(s,\tau) := \mathbb{E}\left(e^{s\left(\hat{L}_1 - (\mathbb{E}(L_2(2)) - \tau)\right)}\right)$, it is easy to see that $p(s,\tau)$ is smooth in s, and that

$$\begin{cases} p(0,\tau) = 1, \\ \frac{\partial p(s,\tau)}{\partial s}|_{s=0} = \tau > 0, \end{cases}$$

for every $\tau > 0$. Hence,

$$\inf_{s<0} p(s,\tau) < e^{-c(\tau)}, \tag{4.28}$$

for a suitable $c(\tau) > 0$. Thus,

$$\mathbb{P}((H_k^{(n)})^c) \leq \mathbb{P}(L_k(k) < \xi_m k) \leq \mathbb{P}\left(\sum_{i=1}^{k/2} \hat{L}_i < k\left(\frac{\mathbb{E}(\hat{L}_i) - \tau}{2}\right)\right) < e^{-c(\tau)k/2}.$$

Now, let $\tau = \tau_m := \mathbb{E}(L_2(2)) - 2\xi_m$, let $\xi_m = 11/10m$, and so

$$p(s,\tau_m) = \frac{e^{-11s/5m}\left(me^{2s} + (4m^2 - 7m + 3)e^s + m^3 - 4m^2 + 6m - 3\right)}{m^3}.$$

Since $\inf_{s<0} p(s,\tau_m) < e^{-1/1000m}$, one can choose $c(\tau_m) = 1/1000m$. Hence,

$$\mathbb{P}((H^{(n)})^c) \leq \sum_{k=\nu n}^{n} e^{-c(\tau_m)k/2} = \frac{e^{c(\tau_m)(1-n\nu)/2} - e^{-c(\tau_m)n/2}}{e^{c(\tau_m)/2} - 1} \leq \frac{e^{c(\tau_m)/2}}{e^{c(\tau_m)/2} - 1} e^{c(\tau_m)(-n\nu)/2}.$$

Choosing $C_5 = e^{c(\tau_m)/2} \Big/ (e^{c(\tau_m)/2} - 1)$, and $C_6 = c(\tau_m)(\nu)/2$, we have,

$$\mathbb{P}(H^{(n)}) \geq 1 - C_5 e^{-C_6 n}.$$

□

We now finish our argument showing that, with high probability, any $(\pi, \eta) \in M_{min}^k$ contains a non-vanishing proportion $\epsilon > 0$ of unmatched letters. To do so, let

$$I_k^{(n)} := \{L_n(k) \leq (1-\epsilon)\eta(L_n(k)), \text{ for } (\pi, \eta) \in M_{min}^k\},$$

be the event that any pair of matching subsequences $(\pi, \eta) \in M_{min}^k$ has a proportion at least ϵ of unmatched letters, and let

$$I^{(n)} := \bigcap_{k=\nu n}^{n} I_k^{(n)}.$$

Above, $\eta(L_n(k)) - L_n(k)$ is the number of unmatched letters, since $\eta(L_n(k))$ is the position of the last matched letter, while $L_n(k)$ is the number of matched letters.

Lemma 4.10 *Let $\epsilon > 0$ be small enough such that $\delta(\epsilon)$, as given in* (4.24)*, satisfies*

$$\frac{1}{1-\delta(\epsilon)} < \xi_m m, \tag{4.29}$$

where ξ_m is as in (4.25)*. Then, for all $k \geq \nu n$,*

$$G^{(n)}(\epsilon) \cap H_k^{(n)} \subset I_k^{(n)}, \tag{4.30}$$

and thus

$$G^{(n)}(\epsilon) \cap H^{(n)} \subset I^{(n)}. \tag{4.31}$$

Proof. Let $k \in [\nu n, n]$. In order to prove (4.30), we show that if $I_k^{(n)}$ does not hold while $G^{(n)}(\epsilon)$ does hold, then $H_k^{(n)}$ does not hold either. Let $(\pi, \eta) \in M_{min}^k$. If $I_k^{(n)}$ does not hold,

than the proportion of unmatched letters of (π, η) is smaller than ϵ, i.e.,

$$\frac{L_\ell(k)}{\ell} \geq 1 - \epsilon,$$

where $\ell := \eta(L_n(k))$. (Note that $L_\ell(k) = L_n(k)$, since (π, η) is of maximal length.) Therefore,

$$L_\ell(k) \geq \ell(1 - \epsilon). \tag{4.32}$$

Now, when $G_\ell^{(n)}(\epsilon)$ holds, then

$$L_\ell(ml(1 - \delta(\epsilon))) < \ell(1 - \epsilon). \tag{4.33}$$

Comparing (4.32) with (4.33) and noting that the (random) map $x \mapsto L_\ell(x)$ is increasing, yield

$$k \geq m\ell(1 - \delta(\epsilon)),$$

and thus

$$k \geq m\eta(L_n(k))(1 - \delta(\epsilon)) \geq mL_n(k)(1 - \delta(\epsilon)).$$

Hence, from (4.29),

$$\frac{L_n(k)}{k} \leq \frac{1}{m(1 - \delta(\epsilon))} < \xi_m,$$

which implies that $H_k^{(n)}$ cannot hold. □

As an example, when $\epsilon \leq e^{-9}/(1+\ln m)$,

$$\begin{aligned}\delta(\epsilon) &= \epsilon + \sqrt{\frac{2}{C_2}\left(\epsilon(\ln m - \ln \epsilon) - (1-\epsilon)\ln(1-\epsilon)\right)} \\ &\leq \epsilon + 2\sqrt{(1+\ln m - \ln \epsilon)\epsilon} \\ &\leq e^{-9} + 2\sqrt{10e^{-9}} \\ &< \frac{1}{11},\end{aligned}$$

and therefore,

$$\frac{1}{1-\delta(\epsilon)} < \frac{10}{11} = \xi_m m.$$

In order to estimate the event $F^{(n)}$, we need to show that the unmatched letters of $\boldsymbol{Y}^{(n)}$ do not concentrate in a small number of matches of $(\pi,\eta) \in M^k_{min}$. From the minimality of M^k_{min}, the unmatched letters of a match of $(\pi,\eta) \in M^k_{min}$ contain at most one compartment.

Let N^D be the total number of letters in the sequence $\boldsymbol{Y}^{(n)}$ contained in a compartment of length at least D, and let,

$$J^{(n)} := \{N^D \leq \xi_m \epsilon \nu n/2\},$$

where again ξ_m is given via (4.25).

Lemma 4.11 *For any $0 < \epsilon < 1$, there exist a positive integer D, and positive constant C_7 and C_8 depending on D, such that*

$$\mathbb{P}(J^{(n)}) \geq 1 - C_7 e^{-C_8 n}. \tag{4.34}$$

Proof. Let $\tilde{N}^D$ be the number of integers $s \in [0, n-D]$ such that

$$(Y_s, Y_{s+1}, \ldots, Y_{s+D-1}) \text{ belongs to a compartment.} \tag{4.35}$$

It is easy to check that

$$N^D \leq D\tilde{N}^D. \tag{4.36}$$

Let now $\tilde{Y}_s$, $s \in [0, n-D]$, be equal to 1 if and only if (4.35) holds, and 0 otherwise. Clearly,

$$\sum_{s=1}^{n} \tilde{Y}_s = \tilde{N}^D. \tag{4.37}$$

To estimate the sum (4.37), decompose it into D subsums of i.i.d. random variables $\Sigma_1, \Sigma_2, \ldots, \Sigma_D$ where

$$\Sigma_i = \sum_{\substack{s=1,\ldots,n \\ s \bmod D = i}} \tilde{Y}_s,$$

so that

$$\tilde{N}^D = \sum_{i=1}^{D} \Sigma_i. \tag{4.38}$$

Then, from (4.36)

$$\mathbb{P}\left(N^D > \frac{\xi_m \epsilon \nu}{2} n\right) \leq \mathbb{P}\left(\tilde{N}^D > \frac{\xi_m \epsilon \nu}{2D} n\right) \leq D\mathbb{P}\left(\Sigma_1 > \frac{\xi_m \nu \epsilon}{2D^2} n\right), \tag{4.39}$$

since in (4.38) at least one of the summands has to be larger than $n\xi_m \epsilon \nu / 2D^2$. Now, the $\tilde{Y}_s$ appearing in the subsum Σ_1 are i.i.d. Bernoulli random variables with

$$\mathbb{P}(\tilde{Y}_s = 1) \leq m \left(\frac{m-1}{m}\right)^D.$$

Therefore,

$$\mathbb{P}\left(\Sigma_1 > (\mathbb{E}\tilde{Y}_s + \delta)\frac{n}{D}\right) \leq e^{-c(\delta)\frac{n}{D}}, \tag{4.40}$$

with $c(\delta) > 0$ for $\delta > 0$. Take $\delta = \mathbb{P}(\tilde{Y}_s = 0) = 1 - \mathbb{P}(\tilde{Y}_s = 1)$, then $c(\delta) = -\ln \mathbb{P}(\tilde{Y}_s = 1)$. Thus it is enough to choose D such that

$$2Dm\left((m-1)/m\right)^D < \xi_m \nu \epsilon. \tag{4.41}$$

Let $x = (m-1)/m$, $y = \xi_m \nu \epsilon / 2m$, we next show that,

$$D = \frac{1}{y(\ln x)^2} = \frac{40e^9 m^3(1+\ln m)}{11 \ln^2\left(\frac{m-1}{m}\right)}, \tag{4.42}$$

does satisfy (4.41), or equivalently that $Dx^D < y$. With the choice in (4.42), $Dx^D < y$ is equivalent to $2y \ln x \ln y + 2y(\ln x)^2 < 1$, which is true since

$$2y \ln x \ln y + 2y(\ln x)^2 = 2(-\ln x)(-y \ln y) + 2y(\ln x)^2 \leq 2 \ln 2 \cdot 9e^{-9} + 2(\ln 2)^2 e^{-9} < 1.$$

Choosing $C_7 = D$ and $C_8 = c(\delta)/D$, we have

$$\mathbb{P}(J^{(n)}) \geq 1 - C_7 e^{-C_8 n}.$$

□

We can now find a suitable λ such that when $H^{(n)}$, $I^{(n)}$ and $J^{(n)}$ all hold, then $F^{(n)}$ (which depends on λ, see (4.19)) also holds.

Lemma 4.12 *Let ϵ be as in Lemma 4.10, let D be such that $2Dm\left((m-1)/m\right)^D < \xi_m \nu \epsilon$, with*

$$\lambda = \frac{\xi_m \nu}{2} \frac{\epsilon}{D-1}.$$

Then, for $k \geq \nu n$,

$$H^{(n)} \cap J^{(n)} \cap I_k^{(n)} \subset F_k^{(n)}, \tag{4.43}$$

and thus

$$H^{(n)} \cap J^{(n)} \cap I^{(n)} \subset F^{(n)}. \tag{4.44}$$

Proof. We prove (4.43), from which (4.44) immediately follows. On $I_k^{(n)}$, each $(\pi, \eta) \in M_{min}^k$

has at least $\epsilon\eta(L_n(k))$ unmatched letters. But,

$$\eta(L_n(k)) \geq L_n(k). \tag{4.45}$$

When $H^{(n)}$ holds,

$$L_n(k) \geq \xi_m k. \tag{4.46}$$

Since $k \geq \nu n$, (4.45) and (4.46), together imply that the number of unmatched letters of $(\pi, \eta) \in M_{min}^k$ is at least $\epsilon\ \xi_m \nu n$. By $J^{(n)}$, there are at most $\xi_m \nu \epsilon n/2$ letters contained in compartments of length at least D. Thus, there are at least $\xi_m \nu \epsilon n/2$ unmatched letters contained in compartments of length less than D. But, every match of $(\pi, \eta) \in M_{min}^k$ contains unmatched letters from only one compartment, and as such every match can contain at most $D-1$ unmatched letters from compartments of length less than D. Therefore, these $\epsilon\, \xi_m \nu n/2$ unmatched letters which are not in N^D, must fill at least $\epsilon\, \xi_m \nu n/(2D-2)$ matches of $(\pi, \eta) \in M_{min}^k$. Hence, $(\pi, \eta) \in M_{min}^k$ has at least $\epsilon\, \xi_m \nu n/(2D-2)$ non-empty matches. □

Combining Lemma 4.10 and Lemma 4.12 gives,

$$\mathbb{P}((F^{(n)})^c) \leq \mathbb{P}((G^{(n)}(\epsilon))^c) + \mathbb{P}((H^{(n)})^c) + \mathbb{P}((J^{(n)})^c),$$

which via (4.22), (4.26), and (4.34) entails

$$\mathbb{P}(F^{(n)}) \geq 1 - C_4 e^{-C_3 \nu n} - C_5 e^{-C_6 n} - C_7 e^{-C_8 n}.$$

Next, recalling the definition of O_n in (4.6), observe that

$$\mathbb{P}(O_n^c) \leq \mathbb{P}(O_n^c \cap E^{(n)} \cap F^{(n)}) + \mathbb{P}((F^{(n)})^c) + \mathbb{P}((E^{(n)})^c).$$

The next lemma provides an estimate on the first probability, on the above right hand side, and completes the proof of Theorem 4.4.

Lemma 4.13 *Let $K \leq 1/2m$, then*

$$\mathbb{P}(O_n^c \cap E^{(n)} \cap F^{(n)}) \leq ne^{-2K^2 h(n)}.$$

Proof. Let λ given as in Lemma 4.12 be at most 1, and let $K := \lambda/2m$, so that $K \leq 1/2m$. Let

$$\Delta(k) := \begin{cases} L_n(k+1) - L_n(k) & \text{when } F_k^{(n)} \text{ holds,} \\ 1 & \text{otherwise.} \end{cases}$$

From (4.20), it follows that:

$$\mathbb{P}\left(\Delta(k) = 1 | \, \sigma_k\right) \geq \lambda/m, \tag{4.47}$$

where σ_k denote the σ-field generated by the Z_i^k and Y_j, namely,

$$\sigma(Z_i^k, Y_j \mid i \leq k, j \leq n).$$

Moreover, $\Delta(k)$ is equal to zero or one (since $L_n(\cdot)$ is non-decreasing on $\mathbb{N}$) and is also σ_k-measurable. Let

$$\tilde{L}_n(k) = \begin{cases} L_n(\nu n) + \sum_{i=\nu n}^{k-1} \Delta(i) & \text{for } k \in [\nu n, n], \\ L_n(k) & \text{for } k \in [0, \nu n]. \end{cases}$$

Note that when $F^{(n)}$ holds, then

$$L(k) = \tilde{L}(k), \tag{4.48}$$

for all $k \in [0, n-1]$. Define

$$\tilde{O}_{i,j}^{(n)} = \{\tilde{L}_n(j) - \tilde{L}_n(i) \geq K(j-i)\},$$

and

$$\tilde{O}_n = \bigcap_{\substack{i,j \in I \cap [\nu n, n] \\ j \geq i + h(n)}} \tilde{O}_{i,j}^{(n)}.$$

When $E^{(n)}$ holds, then $L_n(k)$ has a slope of one on the domain $[0, \nu n]$. Therefore, since $K \leq 1/2m$, the slope condition of O_n holds on the domain $[0, \nu n] \cap I$. When $F^{(n)}$ holds, then $L_n(k)$ and $\tilde{L}_n(k)$ are equal. Therefore, when $F^{(n)}$ and $\tilde{O}_n$ both hold, then the slope condition of O_n is verified on the domain $[\nu n, n] \cap I$. Hence,

$$E^{(n)} \cap F^{(n)} \cap \tilde{O}_n = E^{(n)} \cap F^{(n)} \cap O_n, \tag{4.49}$$

and thus

$$\mathbb{P}(O_n^c \cap E^{(n)} \cap F^{(n)}) = \mathbb{P}(\tilde{O}_n^c \cap E^{(n)} \cap F^{(n)}) \leq \mathbb{P}(\tilde{O}_n^c).$$

It only remains to estimate $\mathbb{P}(\tilde{O}_n^c)$. First,

$$\mathbb{P}(\tilde{O}_n^c) \leq \sum_{\substack{i,j \in I \cap [\nu n, n] \\ j \geq i + h(n)}} \mathbb{P}((\tilde{O}_{i,j}^{(n)})^c). \tag{4.50}$$

Then, from Hoeffding's exponential inequality, for any $t > 0$,

$$\mathbb{P}\left(\frac{\sum_{s=i}^{j} \Delta(s)}{j - i} < \mathbb{E}\Delta(i) - t \right) < e^{-2(j-i)t^2}. \tag{4.51}$$

With the help of (4.47), and since $K = \lambda/2m$, by choosing $t = \mathbb{E}\Delta(i) - K$, (4.51) becomes

$$\mathbb{P}((\tilde{O}_{i,j}^{(n)})^c) \leq e^{-2|i-j|(\mathbb{E}\Delta(i)-K)^2} \leq e^{-2K^2 h(n)}, \tag{4.52}$$

for all $i, j \in [\nu n, n]$. Then, note that there are at most n terms in the sum in (4.50). Thus (4.50) and (4.52) together imply that

$$\mathbb{P}(\tilde{O}_n^c) \leq n e^{-2K^2 h(n)}. \tag{4.53}$$

□

4.4 Estimation of the Constants

To estimate C in (4.3), we need to first estimate various constants.

First let $\nu = 1/2m$. Next, to estimate K_1, the right hand side of (4.12) needs to be lower bounded. When $n \geq 900/(p(1-p))$, (4.17) gives that

$$\mathrm{Var}(N|\mathbf{1}_{N\in I} = 1) \geq \frac{1}{1000}p(1-p)n.$$

Therefore, any K_1 satisfying $0 < K_1 < \sqrt{p(1-p)}/(10\sqrt{10})$ is fine. Choosing $K_1 = \sqrt{p(1-p)}/(20\sqrt{5})$, then

$$\mathrm{Var}(N|\mathbf{1}_{N\in I} = 1) - h(n)^2 \geq \frac{1}{2000}p(1-p)n.$$

To estimate A and B in (4.7) requires upper bounds on C_4, C_5, C_7 and lower bounds for C_3, C_6, C_8. As shown after Lemma 4.10, we can choose $\epsilon = e^{-9}/(1+\ln m)$, then

$$\begin{aligned} C_3 &= (\delta(\epsilon) - \epsilon)^2 C_2/2 \\ &= \epsilon \ln m - (1-\epsilon)\ln(1-\epsilon) - \epsilon \ln \epsilon \\ &\geq \epsilon \ln m \geq e^{-10}, \end{aligned}$$

and

$$C_4 = 1/(1 - e^{-C_3}) \leq e^{11}.$$

Lemma 4.9 gives

$$C_5 = \frac{e^{1/2000m}}{e^{1/2000m} - 1} \leq 1 + 2000m,$$

and

$$C_6 = \frac{1}{4000m^2}.$$

Lemma 4.11 gives

$$C_7 = D \leq 20e^9,$$

and

$$\begin{aligned} C_8 &= \frac{c(\delta)}{D} \\ &= \frac{-\ln \mathbb{P}(\tilde{Y}_s = 1)}{D} \\ &\geq \ln \frac{m}{m-1} - \frac{\ln m}{D} \\ &= \ln \frac{m}{m-1} \left(1 - \frac{11 \ln m \ln \frac{m}{m-1}}{40e^9 m^3 (1 + \ln m)}\right) \\ &\geq \ln \frac{m}{m-1} \left(1 - \frac{1}{20e^9}\right) \geq \frac{1}{2m}. \end{aligned}$$

Therefore, one can take $A = \max\{1 + 2000m, 20e^9\}$ and $B = e^{-10}/m^2$. Then, for $n \geq e^{10} m^2 \ln(80e^9 + 8000m)$, $\mathbb{P}(O_n) \geq 1/2$

Note that when $n \geq 900/(p(1-p))$, we also have $\mathbb{P}(N \in I) \geq 1/2$. Let

$$C_9 = \frac{K^2}{64000} p(1-p),$$

and let

$$C_{10} = \min_{n \leq \max\{900/(p(1-p)), e^{10} m^2 \ln(80e^9 + 8000m)\}} \frac{\operatorname{Var} LC_n}{n},$$

then one can choose $C = \min\{C_9, C_{10}\}$ in (4.3).

4.5 Concluding Remarks

- The results of the paper show that we can approach as closely as we want the uniform case and have a linear order on the variance of LC_n. However, the lower order of the variance in the uniform case is still unknown although numerical results, see [22], leave little doubt that the variance is linear in the length of the words. (Unfortunately, the estimates of the previous section, on $C = C(p, m)$ in (4.3), converge to zero as

$p \to 0$.)

- Combining the above results with techniques and results presented in [15], the upper and lower bound obtained above can be generalized to provide estimates of order $n^{r/2}$, $r \geq 1$, on the centered r-th moment of LC_n.

- Finally, the above results can also be extended to the general case where the letters of one sequence are taken with probability p_i, $i = 1, 2, \ldots, m$, where $p_i > 0$ and $\sum_{i=1}^{m} p_i = 1$, while for the other sequence the first m letters are taken with probability $p_i - r_i > 0$ and the extra letter is taken with probability $\sum_{i=1}^{m} r_i$. Then many of the lemmas remain true replacing $1/m$ by $\inf_{i=1,\ldots,m} p_i$ or $\inf_{i=1,\ldots,m}(p_i - r_i)/(1 - \sum_{i=1}^{m} r_i)$. For example, in the heading of Section 4.3.1 and Section 4.3.2, in (4.18) and (4.20), Lemma 4.8, and Lemma 4.13, the $1/m$ can be replaced by $\inf_{i=1,\ldots,m} p_i$. In (4.21) of Lemma 4.6, and in the definition of $G_{\ell}^{(n)}(\epsilon)$ in Lemma 4.7, the term $L_{\ell}(m\ell(1-\delta))$ would have to be replaced with

$$L_{\ell}\left(\frac{\ell(1-\delta)(1-\sum_{i=1}^{m} r_i)}{\inf_{i=1,\ldots,m}(p_i - r_i)}\right).$$

With these adjustments, the final lower bounds will have similar forms.

CHAPTER 5
CONCLUSION

In this dissertation we first studied two state-of-the-art algorithms for computing the longest common subsequences of two words—the WMMM algorithm, which was best for small alphabet sizes, and the Kuo-Cross algorithm, which was best for large alphabet sizes. We reduced the time complexity of the Kuo-Cross algorithm by $\Theta(n \log n)$. Using these two algorithms, we performed Monte Carlo simulations to study the statistical behavior in mean and variance of the length of the longest common subsequences. Specifically, we empirically verified that when n is large, the variance of the length of the LCSs of binary words is of order n. We also used Monte Carlo methods to estimate the expectation and variance of LC_n for alphabets of size four and, in turn, these results are used to develop a hypothesis testing method for sequences similarity. We also derived theoretical upper bounds for the Chvátal-Sankoff constant of multiple sequences. Finally, as a main contribution to this dissertation, we proved that the variance of the length of the longest common subsequences is of linear order when the two words are drawn from certain asymmetrical distributions.

Many problems remain open on the topics of sequences comparisons. For example, the size of the alphabet could be countably infinite. A classical application of such an instance, with infinite alphabet, is the `diff` utility. The `diff` utility is a computer software that uses the LCSs of two files to show how to edit one file into another, with minimal line changes, where each line of a file is considered as a "character" of a sequence. In this case, LCSs are, for example, helpful in comparing two versions of a computer file. Other problems of interest would be to determine the asymptotic behavior of the expectation and variance of the length of LCSs for more than two sequences. These problems may also have applications in areas such as data compression. For example, for multiple similar sequences of DNA, one can use an LCS as a reference sequence and represent other sequences as

small modifications of the reference sequence, thus saving storage spaces. The study of the probabilistic behavior of the length of LCSs might help in estimating the efficiency of such a compression. New questions arise, for totally ordered alphabets, with the study of longest common and increasing subsequences (LCISs) of random permutations or random words. For example, there is an analog of the Chvátal-Sankoff constant for the LCISs of random permutations, which is still unknown, and obtaining any tight estimation or bounds on it would be more than worthwhile. Finally, exploring the connections between the asymptotic laws of these lengths, properly centered and normalized, and eigenvalues of Gaussian random matrices is a fascinating research project for the long run.

Appendices

APPENDIX A

EXPERIMENTAL PLATFORM

For all the simulations presented in this paper, the experiments were run on the Partnership for an Advanced Computing Environment (PACE).

APPENDIX B

CODES

B.1 lcs_common.h

```
#ifndef LCS_COMMON_H
#define LCS_COMMON_H
#include <vector>

typedef int length_type;
typedef char symbol_type;
typedef std::vector<symbol_type> sequence_type;

#endif
```

B.2 lcs_kuocross.h

```
#ifndef LCS_KUOCROSS_H
#define LCS_KUOCROSS_H
#include <vector>
#include "lcs_common.h"

sequence_type::size_type lcs_kuocross_dirty(
  const sequence_type &A,
  const sequence_type &B,
  symbol_type max_symbol,
  std::vector<sequence_type::size_type> &thresh,
```

```
  std::vector<sequence_type::size_type> &matchlist,
  std::vector<sequence_type::size_type> &matchlist_next,
  std::vector<sequence_type::size_type> &sym2Aidx,
  std::vector<sequence_type::size_type> &last_list_idx);

#endif
```

B.3 lcs_kuocross.cc

```
#include "lcs_kuocross.h"
#include <iostream>

void build_matchlist_dirty(
  const sequence_type &A,
  const sequence_type &B,
  symbol_type max_symbol,
  std::vector<sequence_type::size_type> &matchlist,
  std::vector<sequence_type::size_type> &matchlist_next,
  std::vector<sequence_type::size_type> &sym2Aidx,
  std::vector<sequence_type::size_type> &last_list_idx
  )
{
  sequence_type::size_type m = A.size(), n = B.size();
  sym2Aidx.assign(max_symbol+1,m);
  matchlist.assign(m,n);
  matchlist_next.assign(n,n);
  last_list_idx.assign(m,n);
```

```
  for (sequence_type::size_type i=0; i<m; i++) {
    sym2Aidx[A[i]] = i;
  }
  for (sequence_type::size_type j=0; j<n; j++) {
    auto s = B[j];
    auto i = sym2Aidx[s];
    if (i!=m) {
      if (matchlist[i] == n) {
        matchlist[i] = j;
      } else {
        matchlist_next[last_list_idx[i]] = j;
      }
      last_list_idx[i]=j;
    }
  }
  for (sequence_type::size_type i=0; i<m; i++) {
    auto j = sym2Aidx[A[i]];
    if (j != i) {
      matchlist[i] = matchlist[j];
    }
  }
}

sequence_type::size_type lcs_kuocross_dirty(
  const sequence_type &A_in,
  const sequence_type &B_in,
  symbol_type max_symbol,
```

```
  std::vector<sequence_type::size_type> &thresh,
  std::vector<sequence_type::size_type> &matchlist,
  std::vector<sequence_type::size_type> &matchlist_next,
  std::vector<sequence_type::size_type> &sym2Aidx,
  std::vector<sequence_type::size_type> &last_list_idx)
{
  sequence_type::size_type m = A_in.size(), n = B_in.size();
  const sequence_type &A = m>n?B_in:A_in;
  const sequence_type &B = m>n?A_in:B_in;
  if (m > n) {
    std::swap(m,n);
  }

  build_matchlist_dirty(A,B,max_symbol,matchlist,
          matchlist_next,sym2Aidx,last_list_idx);

  thresh.assign(m+1,n+1);
  thresh[0] = 0;

  for (sequence_type::size_type i=0; i<m; i++) {
    sequence_type::size_type temp = 0;
    std::vector<sequence_type::size_type>::size_type k = 1;
    for (auto j=matchlist[i];j!=n;j=matchlist_next[j]) {
      auto l = j+1;
      if (l > temp) {
        while (l > thresh[k]) {
          k++;
```

```
        }
        temp = thresh[k];
        thresh[k] = l;
      }
    }

  }

  for (sequence_type::size_type k=m+1; k-->0 ;) {
    if (thresh[k] != n+1) {
      return k;
    }
  }
  return 0; // Should never reach here.
}
```

B.4 lcs_wmmm.h

```
#ifndef LCS_WMMM_H
#define LCS_WMMM_H
#include <vector>
#include "lcs_common.h"

length_type lcs_wmmm_dirty(const sequence_type &A,
                           const sequence_type &B,
                           std::vector<length_type> &fp);

#endif
```

B.5 lcs_wmmm.cc

```
#include "lcs_wmmm.h"
#include <utility>
#include <algorithm>

length_type snake(const sequence_type &A,
                  const sequence_type &B,
                  length_type m,
                  length_type n,
                  length_type k,
                  length_type y)
{
    length_type x=y-k;
    while (x<m && y<n && A[x]==B[y]) {
        x++;
        y++;
    }
    return y;
}

length_type lcs_wmmm_dirty(const sequence_type &A_in,
                      const sequence_type &B_in,
                      std::vector<length_type> &fp)
{
    length_type m = A_in.size(), n = B_in.size();
    const sequence_type &A = m>n?B_in:A_in;
```

```
const sequence_type &B = m>n?A_in:B_in;
if (m > n) {
    std::swap(m,n);
}

length_type k;
length_type delta = n - m;

length_type size = m + n + 1;
fp.resize(size);
std::fill(fp.begin(),fp.end(),-1);
length_type offset = m;

length_type p = -1;
do {
    p++;
    for (k=-p; k<delta; k++) {
        fp[offset+k] = snake(A,B,m,n,k,
                        std::max(fp[offset+k-1]+1,fp[offset+k+1]));
    }
    for (k=delta+p; k>delta; k--) {
        fp[offset+k] = snake(A,B,m,n,k,
                        std::max(fp[offset+k-1]+1,fp[offset+k+1]));
    }
    fp[offset+delta] = snake(A,B,m,n,delta,
                        std::max(fp[offset+delta-1]+1,fp[offset+delta+1]));
} while(fp[offset+delta]<n);
```

```
    return m-p;
}
```

B.6 lcs_dist.cc

```
#include "lcs_wmmm.h"
#include <random>
#include <functional>
#include <iostream>
#include <fstream>
#include <algorithm>
#include "tclap/CmdLine.h"
#include <string>
#include <sstream>
#include <vector>

int main(int argc, char* argv[])
{
    try {
        TCLAP::CmdLine cmd("", ' ', "1.0");
        TCLAP::ValueArg<std::string> arg_dist("d","distribution",
            "Distribution of symbols",
            false,"0.5,0.5","string");
        cmd.add(arg_dist);
        TCLAP::ValueArg<length_type> arg_length("n","length",
           "Length of random sequences",
           false,1000,"int");
        cmd.add(arg_length);
```

```
TCLAP::ValueArg<length_type> arg_niter("m","niter",
   "Number of experiments",
   false,1000,"int");
cmd.add(arg_niter);
TCLAP::ValueArg<std::string> arg_fout("o","output",
   "Output file name",
   false,"lcs_dist_output.csv","string");
cmd.add(arg_fout);

cmd.parse( argc, argv );
// Experiment settings
std::string dist_str = arg_dist.getValue();
length_type niter = arg_niter.getValue();
length_type n = arg_length.getValue();
std::string fout_name = arg_fout.getValue();

std::vector<double> dist_vec;
std::stringstream ss(dist_str);
std::string item;
while(std::getline(ss,item,',')) {
    dist_vec.push_back(std::stod(item));
}

// Setting up random number generator
std::random_device rd;
std::mt19937 mt(rd());
std::discrete_distribution<symbol_type> dist(dist_vec.begin(),dist_vec.e
```

```
    auto gen = std::bind(dist,std::ref(mt));

    // Setting up output files
    std::ofstream f_out;
    f_out.open(fout_name);

    // Prepare buffers.
    sequence_type A(n),B(n);
    std::vector<length_type> fp;
    fp.reserve(n+n+1);

    f_out << n;
    for (length_type iter=0;iter<niter;iter++) {
        std::generate(A.begin(),A.end(),gen);
        std::generate(B.begin(),B.end(),gen);
        f_out << "," <<lcs_wmmm_dirty(A,B,fp);
        f_out.flush();
    }
    f_out << std::endl;
    f_out.close();
    return 0;
}
catch (TCLAP::ArgException &e) {
    std::cerr << "error: " << e.error() << " for arg "
    << e.argId() << std::endl;
}
```

```
}
```

B.7 ha1.cc

```
#include "lcs_wmmm.h"
#include <random>
#include <functional>
#include <iostream>
#include <fstream>
#include <algorithm>
#include "tclap/CmdLine.h"
#include <string>


int main(int argc, char* argv[])
{
    try {
        TCLAP::CmdLine cmd("", ' ', "1.0");
        TCLAP::ValueArg<unsigned int> arg_absize("k","alphabet-size",
                                                 "Alphabet size",false,4,"int");
        cmd.add(arg_absize);
        TCLAP::ValueArg<length_type> arg_lx("x","length-x",
                                            "Length of X and Y",
                                            false,500,"int");
        cmd.add(arg_lx);
        TCLAP::ValueArg<length_type> arg_lz("z","length-z",
                                            "Length of Z",
                                            false,500,"int");
```

```
cmd.add(arg_lz);
TCLAP::ValueArg<int> arg_niter("m","niter",
                               "Number of experiments",
                               false,1000,"int");
cmd.add(arg_niter);
TCLAP::ValueArg<int> arg_ninsert("i","ninsert",
                               "Number of inserts",
                               false,22,"int");
cmd.add(arg_ninsert);
TCLAP::ValueArg<std::string> arg_fout("o","output",
                               "Output file name",
                               false,"exp4.csv","string");
cmd.add(arg_fout);
cmd.parse( argc, argv );

// Experiment settings
const symbol_type ABSIZE = (symbol_type) arg_absize.getValue(); // Alphai
length_type lA = arg_lx.getValue();
length_type lB = lA;
length_type lP = arg_lz.getValue();
int niter = arg_niter.getValue();
int ninsert = arg_ninsert.getValue();
std::string fout_name = arg_fout.getValue();

std::vector<double> abreaks(ninsert+2);
std::vector<double> pbreaks(ninsert+1);
for (int i = 0; i<=ninsert;i++) {
```

```
    pbreaks[i] = ((double) i)/(ninsert+1);
    abreaks[i] = ((double) i)/(ninsert+2);
}
abreaks[ninsert+1] = 1.0;

// Setting up random number generator
std::random_device rd;
std::mt19937 mt(rd());
std::uniform_int_distribution<symbol_type> dist(0,ABSIZE-1);
auto gen = std::bind(dist,std::ref(mt));

// Setting up output files
std::ofstream fout;
fout.open(fout_name);

// Prepare buffers.
sequence_type A(lA),B(lB),P(lP);
sequence_type AA,BB;
std::vector<length_type> fp;

P.reserve(lP);
AA.reserve(lA+lP);
BB.reserve(lB+lP);
fp.reserve(lA+lB+lP+lP+1);

P.resize(lP);
AA.resize(lA+lP);
```

```
BB.resize(lB+lP);

// Experiments
fout << lA+lP;
for (int iter=0; iter<niter; iter++) {
    std::generate(A.begin(),A.end(),gen);
    std::generate(B.begin(),B.end(),gen);
    std::generate(P.begin(),P.end(),gen);

    auto ita = AA.begin();
    auto itb = BB.begin();
    ita = std::copy(A.begin()+(int)(abreaks[0]*lA),
                    A.begin()+(int)(abreaks[1]*lA),
                    ita);
    itb = std::copy(B.begin()+(int)(abreaks[0]*lB),
                    B.begin()+(int)(abreaks[1]*lB),
                    itb);
    for (int i=0;i<ninsert;i++) {
        ita = std::copy(P.begin()+(int)(pbreaks[i]*lP),
                        P.begin()+(int)(pbreaks[i+1]*lP),
                        ita);
        itb = std::copy(P.begin()+(int)(pbreaks[i]*lP),
                        P.begin()+(int)(pbreaks[i+1]*lP),
                        itb);
        ita = std::copy(A.begin()+(int)(abreaks[i+1]*lA),
                        A.begin()+(int)(abreaks[i+2]*lA),
                        ita);
```

```
                itb = std::copy(B.begin()+(int)(abreaks[i+1]*lB),
                                B.begin()+(int)(abreaks[i+2]*lB),
                                itb);
            }
            fout << "," << lcs_wmmm_dirty(AA,BB,fp);
        }
        fout << std::endl;

        // Clean
        fout.close();
        return 0;
    } catch (TCLAP::ArgException &e) {
        std::cerr << "error: " << e.error() << " for arg "
                  << e.argId() << std::endl;
    }

}
```

B.8 ha2.cc

```
#include "lcs_wmmm.h"
#include <random>
#include <functional>
#include <iostream>
#include <fstream>
#include <algorithm>
#include "tclap/CmdLine.h"
#include <string>
```

```
int main(int argc, char* argv[])
{
    try {
        TCLAP::CmdLine cmd("", ' ', "1.0");
        TCLAP::ValueArg<unsigned int> arg_absize("k","alphabet-size",
                                                 "Alphabet size",false,4,"int");
        cmd.add(arg_absize);
        TCLAP::ValueArg<length_type> arg_lx("x","length-x",
                                            "Length of X and Y",
                                            false,500,"int");
        cmd.add(arg_lx);
        TCLAP::ValueArg<length_type> arg_lz("z","length-z",
                                            "Length of Z",
                                            false,500,"int");
        cmd.add(arg_lz);
        TCLAP::ValueArg<int> arg_niter("m","niter",
                                       "Number of experiments",
                                       false,1000,"int");
        cmd.add(arg_niter);
        TCLAP::ValueArg<int> arg_ninsert("i","ninsert",
                                         "Number of inserts",
                                         false,22,"int");
        cmd.add(arg_ninsert);
        TCLAP::ValueArg<std::string> arg_fout("o","output",
                                              "Output file name",
```

```
                                                false,"exp5.csv","string");
cmd.add(arg_fout);
cmd.parse( argc, argv );

// Experiment settings
const symbol_type ABSIZE = (symbol_type) arg_absize.getValue(); // Alphai
length_type lA = arg_lx.getValue();
length_type lB = lA;
length_type lP = arg_lz.getValue();
int niter = arg_niter.getValue();
int ninsert = arg_ninsert.getValue();
std::string fout_name = arg_fout.getValue();

std::vector<double> abreaks(ninsert+2);
std::vector<double> pbreaks(ninsert+1);
for (int i = 0; i<=ninsert;i++) {
    pbreaks[i] = ((double) i)/(ninsert+1);
    abreaks[i] = ((double) i)/(ninsert+2);
}
abreaks[ninsert+1] = 1.0;

// Setting up random number generator
std::random_device rd;
std::mt19937 mt(rd());
std::uniform_int_distribution<symbol_type> dist(0,ABSIZE-1);
auto gen = std::bind(dist,std::ref(mt));
std::discrete_distribution<int> dist2 {0.1,0.1,0.8};
```

```
// Setting up output files
std::ofstream fout;
fout.open(fout_name);

// Prepare buffers.
sequence_type A(lA),B(lB),P(lP);
sequence_type AA,BB;
std::vector<length_type> fp;

P.reserve(lP);
AA.reserve(lA+lP);
BB.reserve(lB+lP);
fp.reserve(lA+lB+lP+lP+1);

P.resize(lP);

// Experiments
for (int iter=0; iter<niter; iter++) {
    std::generate(A.begin(),A.end(),gen);
    std::generate(B.begin(),B.end(),gen);
    std::generate(P.begin(),P.end(),gen);
    AA.resize(0);
    BB.resize(0);

    std::copy(A.begin()+(int)(abreaks[0]*lA),
              A.begin()+(int)(abreaks[1]*lA),
```

```
                std::back_inserter(AA));
        std::copy(B.begin()+(int)(abreaks[0]*lB),
                B.begin()+(int)(abreaks[1]*lB),
                std::back_inserter(BB));
        for (int i=0;i<ninsert;i++) {
            int choice = dist2(mt)+1;
            if (choice & 1) {
                std::copy(P.begin()+(int)(pbreaks[i]*lP),
                        P.begin()+(int)(pbreaks[i+1]*lP),
                        std::back_inserter(AA));
            }
            std::copy(A.begin()+(int)(abreaks[i+1]*lA),
                    A.begin()+(int)(abreaks[i+2]*lA),
                    std::back_inserter(AA));

            if (choice & 2) {
                std::copy(P.begin()+(int)(pbreaks[i]*lP),
                        P.begin()+(int)(pbreaks[i+1]*lP),
                        std::back_inserter(BB));
            }
            std::copy(B.begin()+(int)(abreaks[i+1]*lB),
                    B.begin()+(int)(abreaks[i+2]*lB),
                    std::back_inserter(BB));

        }
    fout << AA.size() << ","
         << BB.size() << ","
```

```
            << lcs_wmmm_dirty(AA,BB,fp) << std::endl;

        }

        // Clean
        fout.close();
        return 0;
    }
    catch (TCLAP::ArgException &e) {
        std::cerr << "error: " << e.error() << " for arg "
                  << e.argId() << std::endl;
    }

}
```

B.9 ha3.cc

```
#include "lcs_wmmm.h"
#include <random>
#include <functional>
#include <iostream>
#include <fstream>
#include <algorithm>
#include "tclap/CmdLine.h"
#include <string>


int main(int argc, char* argv[])
```

```
{
    try {
        TCLAP::CmdLine cmd("", ' ', "1.0");
        TCLAP::ValueArg<unsigned int> arg_absize("k","alphabet-size",
                                                 "Alphabet size",false,4,"int");
        cmd.add(arg_absize);
        TCLAP::ValueArg<length_type> arg_lx("x","length-x",
                                            "Length of X and Y",
                                            false,500,"int");
        cmd.add(arg_lx);
        TCLAP::ValueArg<length_type> arg_lz("z","length-z",
                                            "Length of Z",
                                            false,500,"int");
        cmd.add(arg_lz);
        TCLAP::ValueArg<int> arg_niter("m","niter",
                                       "Number of experiments",
                                       false,1000,"int");
        cmd.add(arg_niter);
        TCLAP::ValueArg<int> arg_ninsert("i","ninsert",
                                         "Number of inserts",
                                         false,22,"int");
        cmd.add(arg_ninsert);
        TCLAP::ValueArg<std::string> arg_fout("o","output",
                                              "Output file name",
                                              false,"exp5.csv","string");
        cmd.add(arg_fout);
        cmd.parse( argc, argv );
```

```
// Experiment settings
const symbol_type ABSIZE = (symbol_type) arg_absize.getValue(); // Alphai
length_type lA = arg_lx.getValue();
length_type lB = lA;
length_type lP = arg_lz.getValue();
int niter = arg_niter.getValue();
int ninsert = arg_ninsert.getValue();
std::string fout_name = arg_fout.getValue();

std::vector<double> abreaks(ninsert+2);
std::vector<double> pbreaks(ninsert+1);
for (int i = 0; i<=ninsert;i++) {
    pbreaks[i] = ((double) i)/(ninsert+1);
    abreaks[i] = ((double) i)/(ninsert+2);
}
abreaks[ninsert+1] = 1.0;

// Setting up random number generator
std::random_device rd;
std::mt19937 mt(rd());
std::uniform_int_distribution<symbol_type> dist(0,ABSIZE-1);
auto gen = std::bind(dist,std::ref(mt));
std::discrete_distribution<int> dist2 {0.05,0.4,0.4,0.15};

// Setting up output files
std::ofstream fout;
```

```
fout.open(fout_name);

// Prepare buffers.
sequence_type A(lA),B(lB),P(lP);
sequence_type AA,BB;
std::vector<length_type> fp;

P.reserve(lP);
AA.reserve(lA+lP);
BB.reserve(lB+lP);
fp.reserve(lA+lB+lP+lP+1);

P.resize(lP);

// Experiments
fout << lA+lP;
for (int iter=0; iter<niter; iter++) {
    std::generate(A.begin(),A.end(),gen);
    std::generate(B.begin(),B.end(),gen);
    std::generate(P.begin(),P.end(),gen);
    AA.resize(0);
    BB.resize(0);

    std::copy(A.begin()+(int)(abreaks[0]*lA),
              A.begin()+(int)(abreaks[1]*lA),
              std::back_inserter(AA));
    std::copy(B.begin()+(int)(abreaks[0]*lB),
```

```
                B.begin()+(int)(abreaks[1]*lB),
                std::back_inserter(BB));
        for (int i=0;i<ninsert;i++) {
            int choice = dist2(mt);
            if (choice & 1) {
                std::copy(P.begin()+(int)(pbreaks[i]*lP),
                          P.begin()+(int)(pbreaks[i+1]*lP),
                          std::back_inserter(AA));
            }
            std::copy(A.begin()+(int)(abreaks[i+1]*lA),
                      A.begin()+(int)(abreaks[i+2]*lA),
                      std::back_inserter(AA));

            if (choice & 2) {
                std::copy(P.begin()+(int)(pbreaks[i]*lP),
                          P.begin()+(int)(pbreaks[i+1]*lP),
                          std::back_inserter(BB));
            }
            std::copy(B.begin()+(int)(abreaks[i+1]*lB),
                      B.begin()+(int)(abreaks[i+2]*lB),
                      std::back_inserter(BB));

        }
    fout << AA.size() << ","
         << BB.size() << ","
         << lcs_wmmm_dirty(AA,BB,fp) << std::endl;
```

```
        }


        // Clean
        fout.close();
        return 0;
    }
    catch (TCLAP::ArgException &e) {
        std::cerr << "error: " << e.error() << " for arg "
                  << e.argId() << std::endl;
    }

}
```

REFERENCES

[1] S. Amsalu, C. Houdré, and H. Matzinger, "Sparse long blocks and the variance of the longest common subsequences in random words," *arXiv:1204.1009v2 [math-ph]*, Sep. 2016. arXiv: 1204.1009 [math-ph].

[2] L. Bareš, "Algorithms for Longest Common Subsequence Computation," Master Thesis, Czech Technical University in Prague, 2009.

[3] F. Bonetto and H. Matzinger, "Fluctuations of the longest common subsequence in the asymmetric case of 2- and 3-letter alphabets," *Latin American Journal of Probability and Mathematical Statistics*, vol. 2, pp. 195–216, 2006.

[4] J. Boutet de Monvel, "Extensive simulations for longest common subsequences," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 7, no. 2, pp. 293–308, 1999.

[5] V. Chvátal and D. Sankoff, "Longest Common Subsequences of Two Random Sequences," *Journal of Applied Probability*, vol. 12, no. 2, pp. 306–315, 1975.

[6] V. Chvátal and D. Sankoff, "An upper-bound technique for lengths of common subsequences," in *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, D. Sankoff and J. Kruskal, Eds., Reading, Massachusetts: Addison-Wesley, 1983.

[7] V. Dancík, "Expected Length of Longest Common Subsequences," PhD thesis, 1994.

[8] J. G. Deken, "Some limit results for longest common subsequences," *Discrete Mathematics*, vol. 26, no. 1, pp. 17–31, Jan. 1979.

[9] J. G. Deken, "Probabilistic behavior of longest-common-subsequence length," in *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, D. Sankoff and J. Kruskal, Eds., Reading, Massachusetts: Addison-Wesley, 1983.

[10] M. Fekete, "Über die Verteilung der Wurzeln bei gewissen algebraischen Gleichungen mit ganzzahligen Koeffizienten," *Mathematische Zeitschrift*, vol. 17, pp. 228–249, 1923.

[11] R. Gong, C. Houdré, and J. Lember, "Lower Bounds on the Generalized Central Moments of the Optimal Alignments Score of Random Sequences," *Journal of Theoretical Probability*, pp. 1–41, Dec. 2016.

[12] D. S. Hirschberg, "A Linear Space Algorithm for Computing Maximal Common Subsequences," *Commun. ACM*, vol. 18, no. 6, pp. 341–343, Jun. 1975.

[13] C. Houdré and Q. Liu, "On the Variance of the Length of the Longest Common Subsequences in Random Words With an Omitted Letter," *To appear in ArXiv*, 2018.

[14] C. Houdré and Ü. Işlak, "A Central Limit Theorem for the Length of the Longest Common Subsequences in Random Words," *arXiv:1408.1559v4 [math]*, Jan. 2017. arXiv: 1408.1559 [math].

[15] C. Houdré and J. Ma, "On the Order of the Central Moments of the Length of the Longest Common Subsequences in Random Words," in *High Dimensional Probability VII*, Cham: Birkhäuser, 2016, pp. 105–136.

[16] C. Houdré and H. Matzinger, "On the Variance of the Optimal Alignments Score for Binary Random Words and an Asymmetric Scoring Function," *Journal of Statistical Physics*, vol. 164, no. 3, pp. 693–734, Aug. 2016.

[17] J. W. Hunt and T. G. Szymanski, "A fast algorithm for computing longest common subsequences," *Commun. ACM*, vol. 20, no. 5, pp. 350–353, May 1977.

[18] M. Kiwi and J. Soto, "On a Speculated Relation Between Chvátal–Sankoff Constants of Several Sequences," *Combinatorics, Probability and Computing*, vol. 18, no. 04, pp. 517–532, Jul. 2009.

[19] S. Kuo and G. R. Cross, "An Improved Algorithm to Find the Length of the Longest Common Subsequence of Two Strings," *SIGIR Forum*, vol. 23, no. 3-4, pp. 89–99, Apr. 1989.

[20] J. Lember and H. Matzinger, "Standard deviation of the longest common subsequence," *The Annals of Probability*, vol. 37, no. 3, pp. 1192–1235, May 2009.

[21] J. Lember, H. Matzinger, J. Sova, and F. Zucca, "Lower bounds for moments of global scores of pairwise Markov chains," *arXiv:1602.05560 [math]*, Feb. 2016. arXiv: 1602.05560 [math].

[22] Q. Liu and C. Houdré, "Simulations, Computations, and Statistics for Longest Common Subsequences," *ArXiv e-prints*, May 2017. arXiv: 1705.06826 [math.PR].

[23] G. S. Lueker, "Improved Bounds on the Average Length of Longest Common Subsequences," *J. ACM*, vol. 56, no. 3, 17:1–17:38, May 2009.

[24] K. Ning and K. P. Choi, "Systematic assessment of the expected length, variance and distribution of Longest Common Subsequences," *arXiv:1306.4253 [cs]*, Jun. 2013. arXiv: 1306.4253 [cs].

[25] J. G. Reich, H. Drabsch, and A. Däumler, "On the statistical assessment of similarities in DNA sequences.," *Nucleic Acids Research*, vol. 12, no. 13, pp. 5529–5543, Jul. 1984.

[26] J. M. Steele, "An Efron-Stein Inequality for Nonsymmetric Statistics," *The Annals of Statistics*, vol. 14, no. 2, pp. 753–758, Jun. 1986.

[27] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *J. ACM*, vol. 21, no. 1, pp. 168–173, Jan. 1974.

[28] S. Wu, U. Manber, G. Myers, and W. Miller, "An O(NP) sequence comparison algorithm," *Information Processing Letters*, vol. 35, no. 6, pp. 317–323, Sep. 1990.